Document Title	Technical Report on Supplementary Material of Workflow Example
Document Category	Technical report (TR)
Document UID	1145
Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R26-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2026-11-26	R26-11	AUTOSAR Release Management	• Implement second iteration including gateway for CAN / ETH
2025-11-27	R25-11	AUTOSAR Release Management	• Initial release

Disclaimer

This work (specification and/or software implementation) and the material contained in it, as released by AUTOSAR, is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the work.

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

This work may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the work may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1 Introduction 4
1.1 Objectives . . . 4
1.2 Scope . . . 4
2 Example System 6
2.1 Architectural Aspects . . . 10
2.2 Technical Aspects . . . 14
2.2.1 Data Exchange Aspects . . . 14
2.2.2 Network Aspects . . . 16
2.2.2.1 CAN . . . 17
2.2.2.2 Ethernet . . . 19
2.2.2.3 Gateway . . . 20
2.3 Differences to Previous Releases . . . 22
3 Related Documentation 23
A Delivered artefacts 24
B Mentioned Class Tables 26

1 Introduction

This technical report provides a comprehensive description of the AUTOSAR Workflow Example. It is designed to enable current and future AUTOSAR partners to experience the complete development process via a well-defined, practical application, using off-the-shelf development tools and standard implementations.

1.1 Objectives

The objectives for the Workflow Example are:

- AUTOSAR is committed to enhance user experience by providing practical guides and examples to ease the entry into the work with AUTOSAR and to show typical applications.
- Since AUTOSAR specifies exchange artefacts, we want to provide practical examples based on these standards that can be processed by as many implementation providers as possible.
- To verify the usability and quality of these exchange artefacts, the processed artefacts must be integrable and systemically functional after deployment to a hardware target.
- This kind of test corresponds to the idea of a plug-fest, in which different providers plug their products installed on a hardware together and provide proof that they can work together.

1.2 Scope

The basis of the Workflow Example is an iterative process which shall handle the challenges of AUTOSAR projects, e.g.

- specifications still leave room for interpretation,
- incomplete implementations of the standard,
- errors that unfortunately creep-in from time to time.

The workflow example provides a system description from which development partners can generate the system extracts for the individual sub-systems in such a way that as many partners as possible can process them smoothly with their tools.

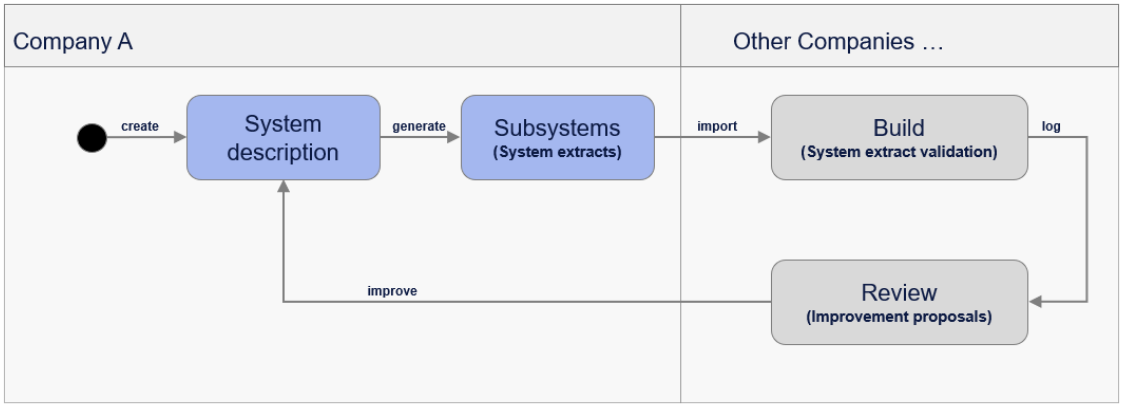


Figure 1.1: Flow of work

2 Example System

The basis function of the system is that a wheel speed which is simulated by a potentiometer, converted by an ADC module (Analog-Digital-Converter) of the WheelSpeed ECU, transmitted via CAN to the ZonalGateway. The `Gateway` connects the CAN bus with the Ethernet bus and transmits the signal to the `HPC`[1] ECU. The `HPC` ECU calculates the vehicle speed based on the receiving signal and sets two commands (control state) according to the vehicle speed, see table in figure 2.2. Afterwards these commands are transmitted to the gateway. The VehicleSpeed ECU receives these commands via CAN from the gateway and indicates the values by two LEDs by a DIO module (Digital-Input-Output).

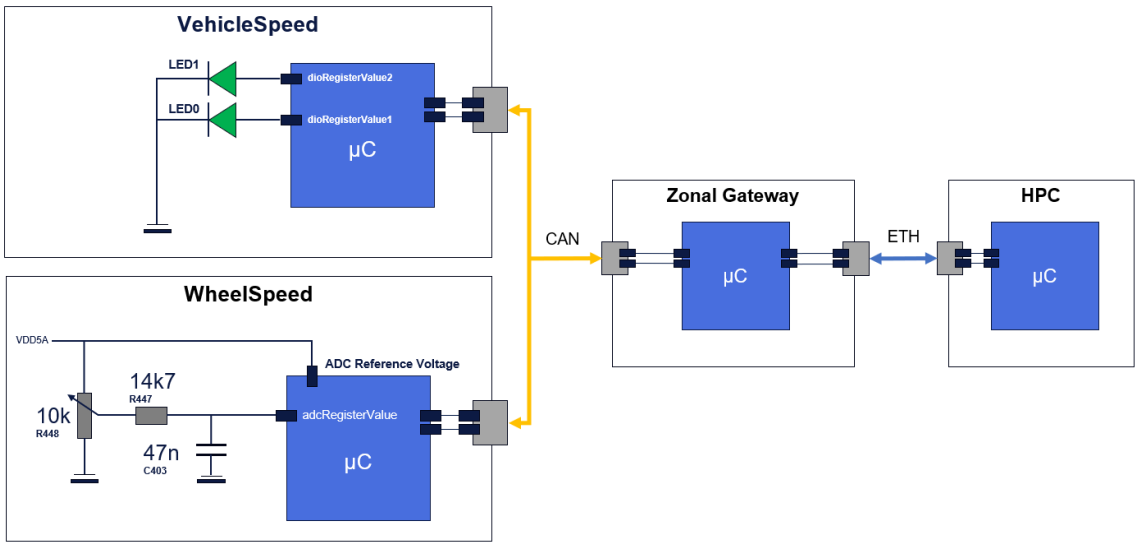


Figure 2.1: Functional sketch of the system

The ZonalGateway and the `HPC` ECU were introduced in iteration two (R26-11) to enable future use cases using AUTOSAR Adaptive features. With the ZonalGateway two bus topologies are used and the `HPC` ECU, still a micro-controller based ECU, opens the perspective to switch to micro-processor based ECU in the next iteration. This implies the application of signal/service translation on the Adaptive Platform side (when in future releases the example is actually implementing an Adaptive Platform Machine as the `HPC`). Signal/service translation is part of the Adaptive Platform Communication Management functional cluster and will be configured accordingly in the next iteration of this example. Transferred in the AUTOSAR methodology the function is defined by five software components in one system composition. Two `SensorActuatorSwComponentType`s for the interaction with outer world, one `SwComponentType` for the internal calculation and two `EcuAbstractionSwComponentType` for the abstraction of the hardware. The `System` class is used to describe the System Configuration of

[1]The abbreviation `HPC` stands for High Performance Computer and is associated with the Adaptive Platform of AUTOSAR. It is realized with a micro-processor based device. In the current iteration (second) of the AUTOSAR Workflow Example, the `HPC` abbreviation is used in the following for a micro-controller based device which will be replaced in the successive iteration of AUTOSAR Workflow Example by a micro-processor based device. Read it as a look ahead.

the complete AUTOSAR system. The category property of System shall indicate the role of this work product. Here the category SYSTEM_DESCRIPTION is used. See [TPS_SYST_01003] in [1]. This is typically done by an OEM.

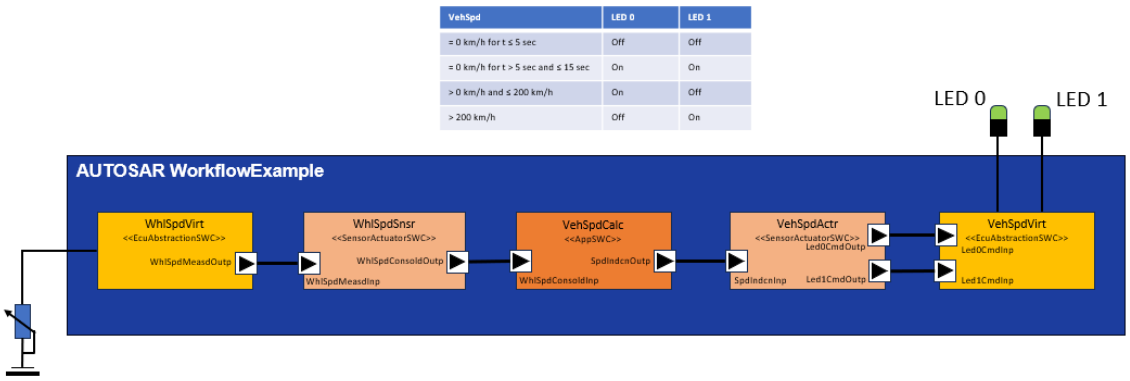


Figure 2.2: Functional representation in AUTOSAR

For the Workflow Example the overall system composition is separated in four SYSTEM_EXTRACTs assigned each to a separate ECU. A distinction between the SYSTEM_EXTRACTs is necessary because the SYSTEM_EXTRACT for the Gateway provides data which realize the transition between the CAN and Ethernet bus. It has no functional content in the sense of an application. This results in three SYSTEM_EXTRACTs (SystemExtract 1, 2 and 3) which are provided to the different suppliers and one SYSTEM_EXTRACT (SystemExtract 4) which is provided to the supplier of the Gateway.

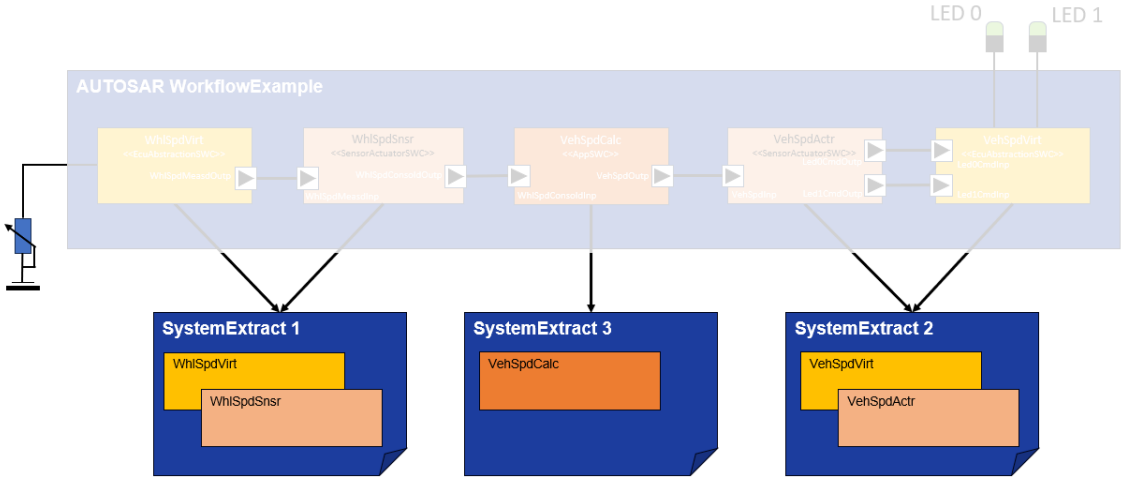


Figure 2.3: Separation in three System Extracts

On the supplier side the SYSTEM_EXTRACT is imported and all required data are added to generate a sub-system for an ECU which will be later integrated in the complete system by the original requester.

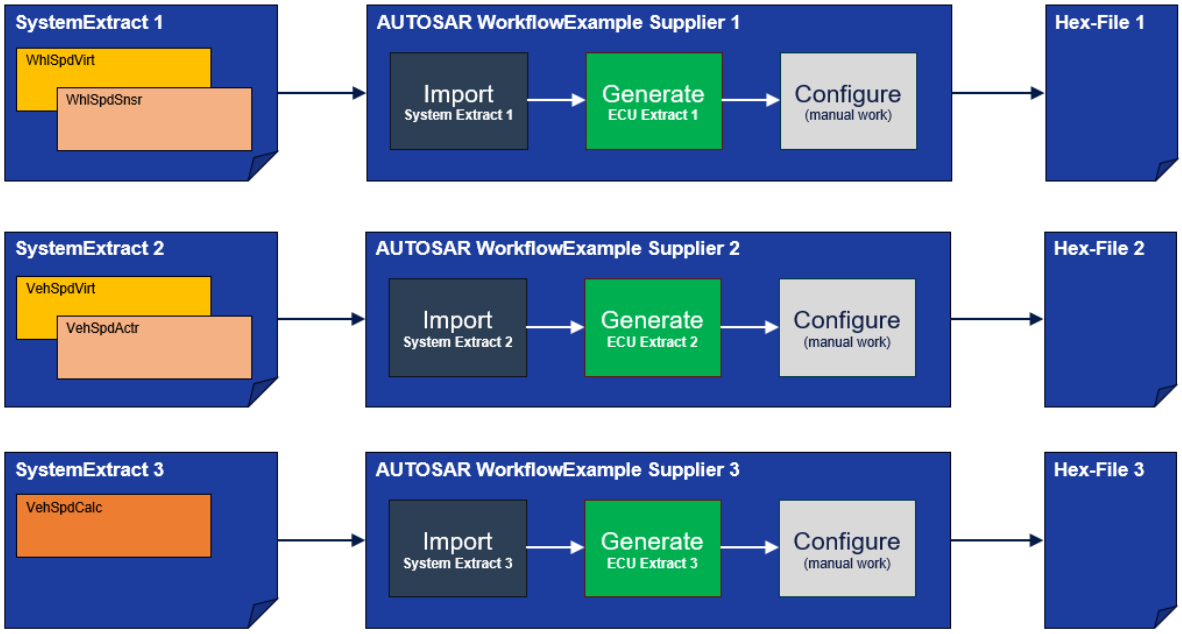


Figure 2.4: Processing of System Extracts on supplier side

Due to the introduction of the additional Ethernet bus, using the ZonalGateway, the hardware topology overview is extended by the `EthernetCluster`.

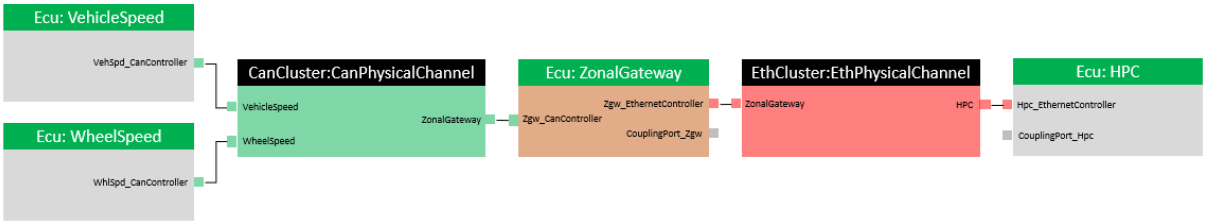


Figure 2.5: Overview of Hardware Topology

As previous mentioned, in addition to the three `SYSTEM_EXTRACT`s (WheelSpeed, VehicleSpeed, `HPC`) illustrated in figure 2.4, a separate `SYSTEM_EXTRACT` for the ZonalGateway is required. Based on this `SYSTEM_EXTRACT` the ZonalGateway implementation can be done. Technical details regarding to the network aspects are described in 2.2.2. The provided `SYSTEM_EXTRACT` contains several methodological aspects defined by the different **T**em**P**late **S**pecifications (TPS documents).

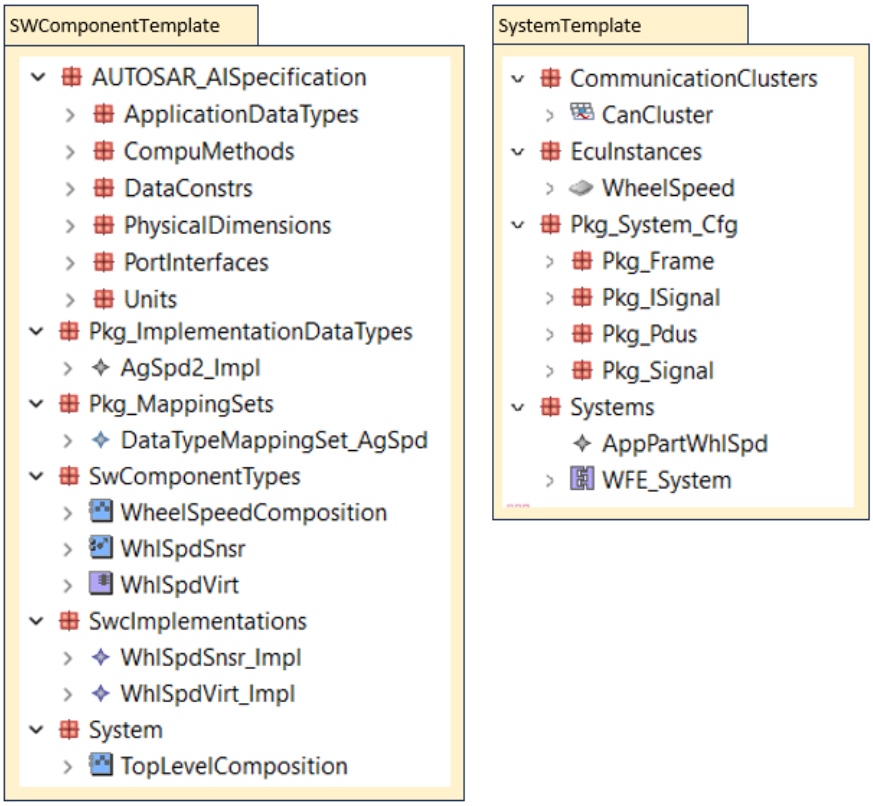


Figure 2.6: Assignment of System Extract artefacts to dedicated Template Specifications applied to WheelSpeed ECU

The packaging of the artifacts is described in `GenericStructure` [2]. A package may contain an arbitrary number of elements, The packaging of the artefacts is described in `GenericStructure` [2]. A package may contain an arbitrary number of elements, represented by the abstract `ARElement`. Note that the aggregation of `ARElement` in `ARPackage` is subject to variation. Therefore the illustration in 2.6 is only one possible representation.

The definition of artefacts of the `SYSTEM_EXTRACT` are provided by the `SWComponentTemplate` [3] and the `SystemTemplate` [1].

In the last step, the ECUs provided by the different suppliers, will be integrated in the target system. The ECUs are connected by the CAN (yellow solid line) and Ethernet (blue solid line) networks.

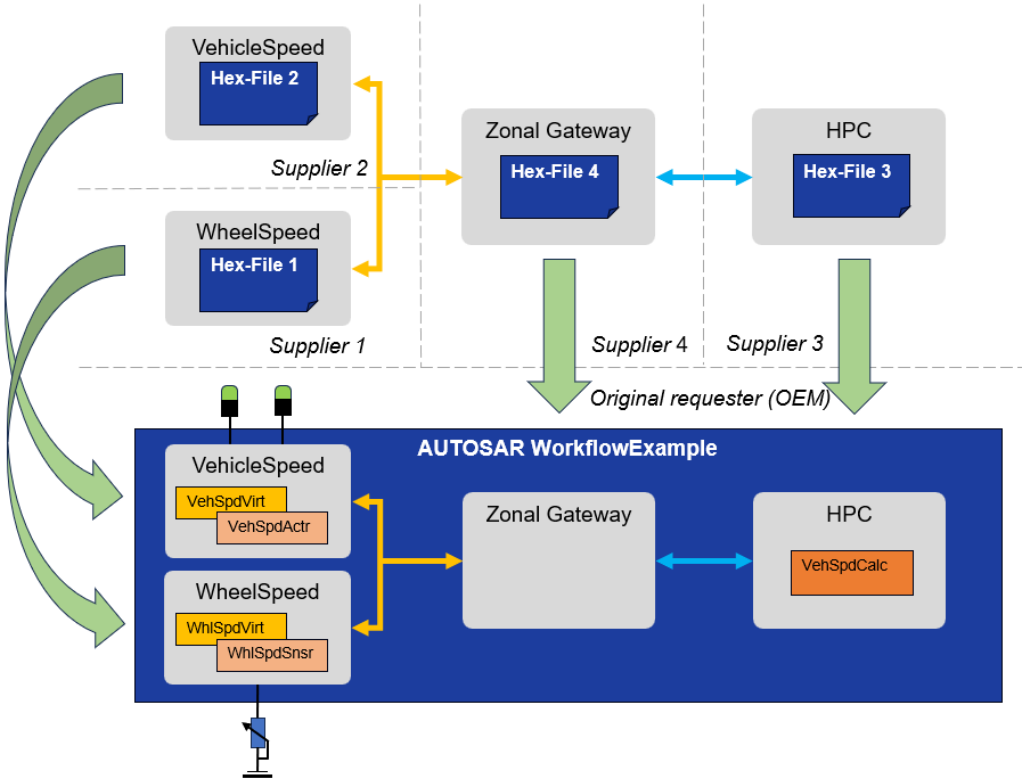


Figure 2.7: Provided ECUs are integrated in target system

2.1 Architectural Aspects

A key conceptual aspect of AUTOSAR is the use of the VFB (Virtual Function Bus). Based on `VirtualFunctionalBus` [4] the Workflow Example is illustrated in figure 2.8.

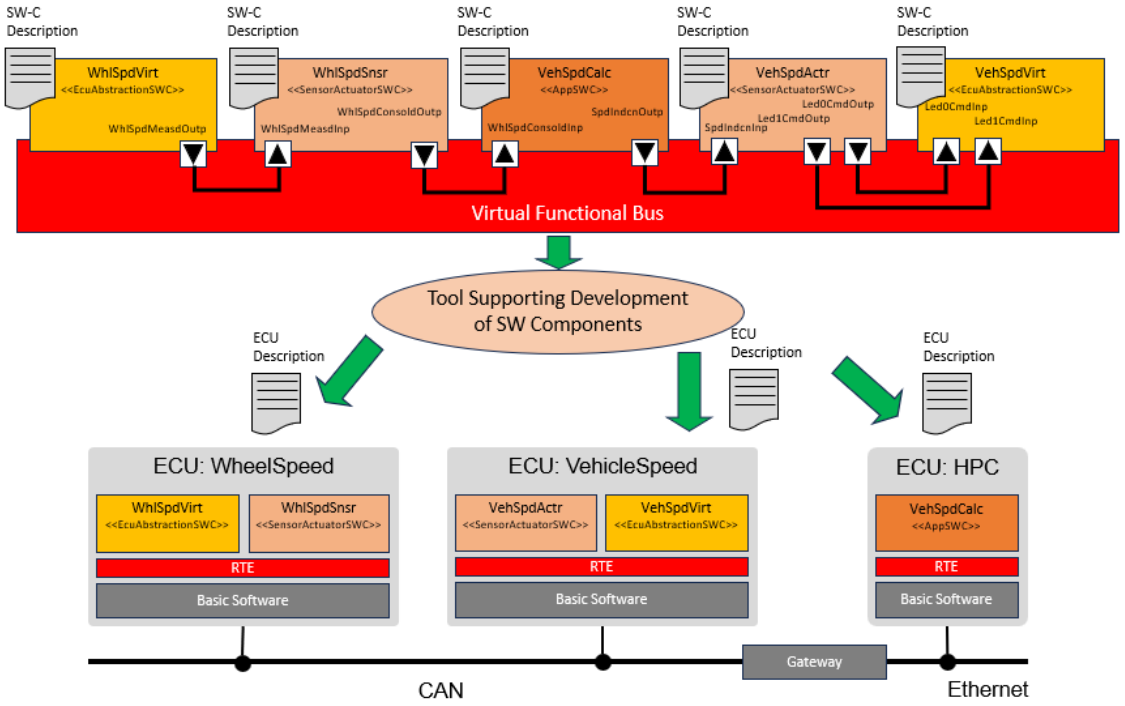


Figure 2.8: Concept of the Virtual Functional Bus

The interconnected software components of the Workflow Example are composed and the Virtual Function Bus is the communication mechanism realizing their interaction (top half of figure 2.8). The components are mapped on specific system resources (ECUs). Thereby, the virtual connections between the components are mapped onto local connections (within a single ECU) or on network-technology specific communication mechanisms (such as CAN or Ethernet frames). Finally, the individual ECUs in such a system can be configured. The concrete interface between the individual components and between the components and the `Basic Software (BSW)` [5] [6] is called the `Run-Time Environment (RTE)` [7].

In the second iteration (R26-11) of the Workflow Example the architecture follows the top-down approach [2]. This approach is implemented in this project and will be described in a brief way.

The top-down approach is described in chapter `13.3 SW component inclusion and top level data mapping` [1] and also used in this project.

The system designer of the "TopLevelComposition" can design the system in such a way that

- exchangeability on application side and
- decoupling of data mapping between network architecture and application architecture

[2] In the first iteration (R25-11) of the Workflow Example both approaches, the top-down and the bottom-up, are described. See 2.3

are achievable. He creates the "TopLevelComposition" based on the RootSwCompositionPrototype and references the CompositionSwComponentType as the top-level software composition. The RootSwComposition will usually not contain all atomic software components that are used in a complete VFB [4] system.

Instead, the focus is on the interfaces between the components or whole compositions. The system design could even contain only compositions and their ports. In this example, the components are included as well.

The system designer (e.g. OEM) is primarily interested in the required functionality and the interfaces defining the integration of the software compositions or software components into the system. In this project the system designer embeds the three sub-compositions "CPT_WheelSpeedComposition", "CPT_HPC_Composition" and "CPT_VehicleSpeedComposition" and the required infrastructure as e.g., the AssemblySwConnector, DelegationSwConnector, PPortPrototype and RPortPrototype.

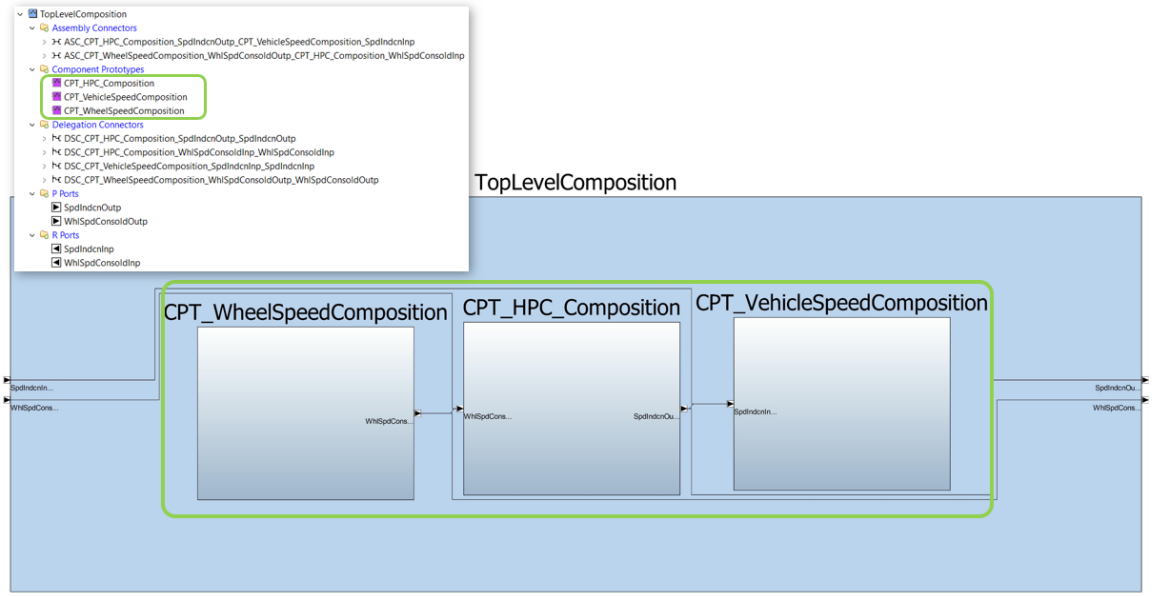


Figure 2.9: Design of TopLevelComposition with three sub-compositions (top-down approach)

An essential aspect of this architectural approach is to decouple the DataMapping of the network communication from that of the implemented applications. The PortPrototypes realizing the network communication are defined at the "TopLevelComposition". The corresponding DataMapping, implemented by the subclass SenderReceiverToSignalMapping, is also defined referring to the outer PortPrototypes, as PPortPrototype and RPortPrototype.

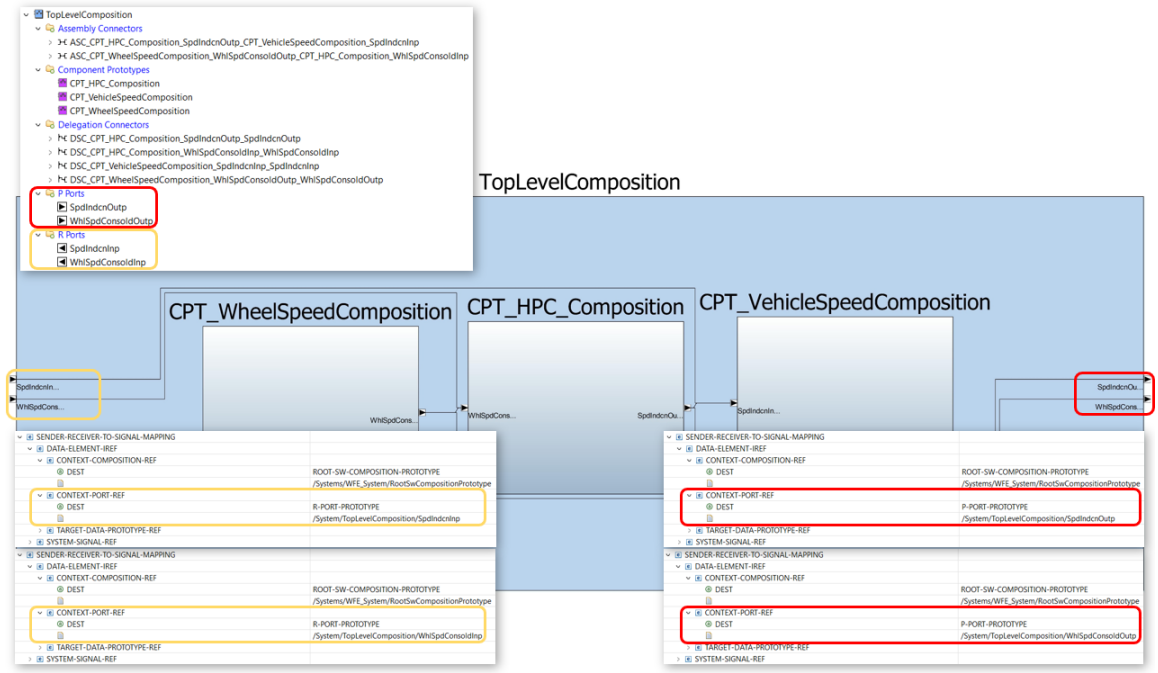


Figure 2.10: Outer `PortPrototype`s realize network communication

From a VFB based point of view the outer `PortPrototype`s are delegated to the inner `PortPrototype`s.

Connection Type	Connector Name	Component Prototype / Composition	Transmit Port		Connected Component Prototype / Composition	Receive Port	
			Port	Port Interface		Port	Port Interface
Delegation	DSC_C	CPT_WheelSpeedComposition	WhlSpdConsoldOutp	RoadWhlAgSpd1	TopLevelComposition	WhlSpdConsoldOutp	RoadWhlAgSpd1
Delegation	DSC_C	CPT_HPC_Composition	SpdIndcnOutp	SpdIndcn	TopLevelComposition	SpdIndcnOutp	SpdIndcn
Delegation	DSC_C	TopLevelComposition	SpdIndcnInp	SpdIndcn	CPT_VehicleSpeedComposition	SpdIndcnInp	SpdIndcn
Delegation	DSC_C	TopLevelComposition	WhlSpdConsoldInp	RoadWhlAgSpd1	CPT_HPC_Composition	WhlSpdConsoldInp	RoadWhlAgSpd1

Figure 2.11: Delegation of outer `PortPrototype`s to inner ones

For the sake of completeness, the inner `PortPrototype`s and assembly connectors are also illustrated.

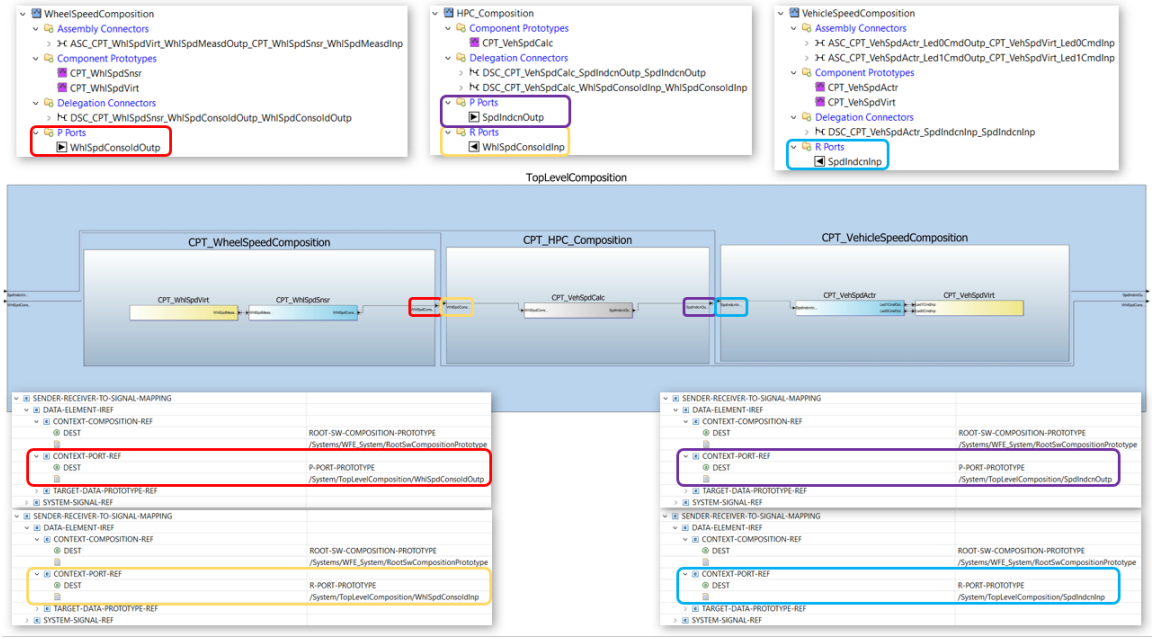


Figure 2.12: Inner `PortPrototype`s realize the communication between the ECUs

Status	Connection Type	Connect or Name	Component Prototype / Composition	Transmit Port		Connected Component Prototype / Composition	Receive Port	
				Port	Port Interface		Port	Port Interface
Valid	Assembly	ASC_CP	CPT_WheelSpeedComposition	WhlSpdConsoldOutp	RoadWhlAgSpd1	CPT_HPC_Composition	WhlSpdConsoldInp	RoadWhlAgSpd1
Valid	Assembly	ASC_CP	CPT_HPC_Composition	SpdIndcnOutp	SpdIndcn	CPT_VehicleSpeedComposition	SpdIndcnInp	SpdIndcn

Figure 2.13: Assembly of inner `PortPrototype`s

The benefit of the described design approach is that actually the software component architecture inside the root composition can be changed without affecting the `DataMapping`s and thus the network definition. Therefore the supplier of the embedded software compositions or software components can act independently.

If the network topology is changed or expanded by the `OEM`, new `SYSTEM_EXTRACT`s must be delivered to the `supplier`s and the corresponding adaptions have to be implemented by them.

2.2 Technical Aspects

2.2.1 Data Exchange Aspects

The data exchange between `OEM` and `supplier` was already illustrated in broad outline in the figures 2.3 and 2.4. Here, the collaboration between both parties, specifically regarding functional data, is examined in greater detail. The system defined in the `SYSTEM_DESCRIPTION` con-

sists of the "TopLevelComposition" containing three sub-compositions.

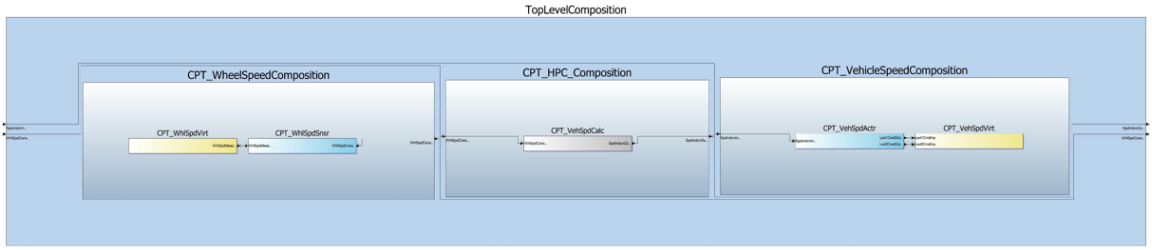


Figure 2.14: TopLevelComposition of designed system

These three sub-compositions are used to generate a `SYSTEM_EXTRACT` for each ECU. The first `SYSTEM_EXTRACT` is dedicated for the "WheelSpeed" ECU.

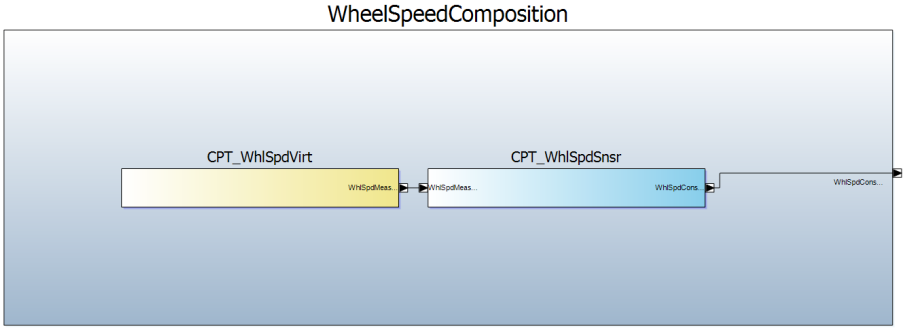


Figure 2.15: Composition of sub-system "WheelSpeed"

This composition consists of the `EcuAbstractionSwComponentType` named "CPT_WhlSpdVirt" which abstracts the hardware and the `SensorActuatorSwComponentType` named "CPT_WhlSpdSnsr" which handles the sensor signal.

The second `SYSTEM_EXTRACT` is dedicated for the "VehicleSpeed" ECU.

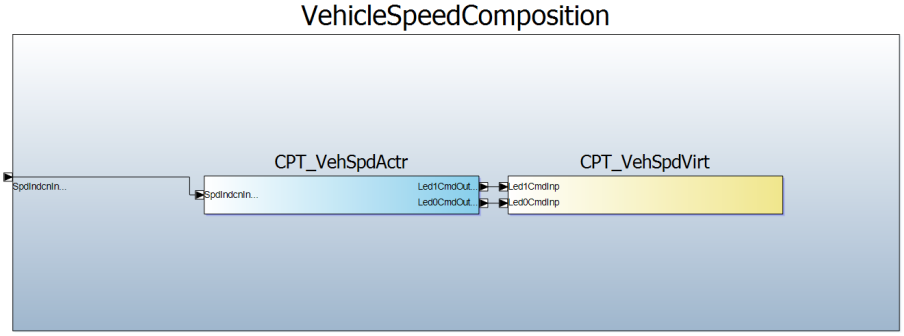


Figure 2.16: Composition of sub-system "VehicleSpeed"

This composition consists of two components: the `SensorActuatorSwComponentType` "CPT_VehSpdActr" prepares the actuator signal for each LED based on

the control state which will be finally received by the `EcuAbstractionSwComponentType` "CPT_VehSpdVirt" which encapsulates the access to the MCAL drivers for the LED.

The third SYSTEM_EXTRACT is dedicated for the "HPC" ECU.

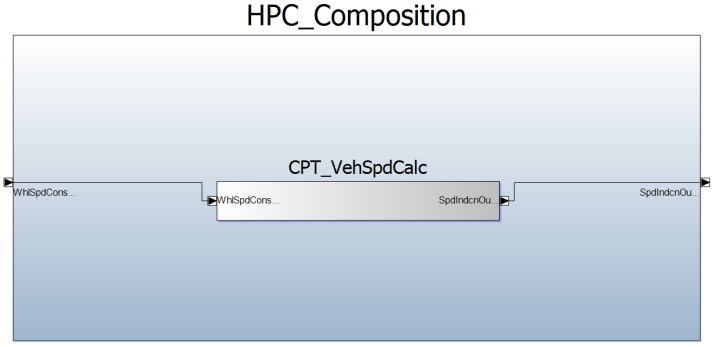


Figure 2.17: Composition of sub-system "HPC"

This composition consists of one component: the ApplicationSwComponentType "CPT_VehSpdCalc" receives the data from the "WheelSpeed" ECU and converts the vehicle speed from the wheel speed. Afterwards, the "CPT_VehSpdCalc" sets 2 LEDs command according to vehicle speed before transmitting to the "VehicleSpeed" ECU. This composition was deliberately created during the second iteration (R26-11) to implement the computational function separately. This enables the accommodation of additional use cases in the future. A significant extension of the implemented system in the second iteration (R26-11) is the use of the Ethernet bus to integrate the third composition (sub-system "HPC"). This integration is facilitated by the Gateway depicted in Figure 2.1. A SYSTEM_EXTRACT is generated for this Gateway as well, although it contains no functional components. The processing of this dedicated SYSTEM_EXTRACT is described in 2.2.2 in detail.

2.2.2 Network Aspects

The system uses two communication buses:

- CAN for the connection between WheelSpeed, VehicleSpeed sensing ECUs and the ZonalGateway ECU.
- Ethernet for the connection between the ZonalGateway ECU and the HPC ECU.

The ZonalGateway ECU bridges both buses by routing I-PDUs between CAN and Ethernet using iPduMapping, see figure 2.1.

Each bus is represented as a CommunicationCluster - the main element grouping all ECUs that share the same communication medium and protocol. A CommunicationCluster carries general attributes such as baudrate, protocolName and aggregates one or more physicalChannel.

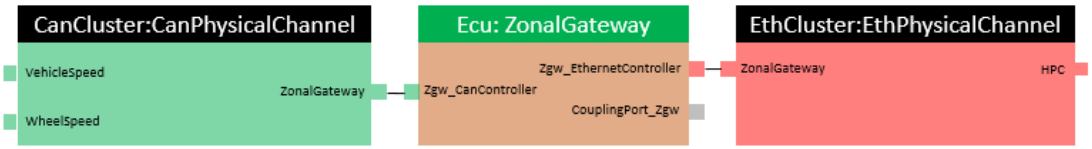


Figure 2.18: `CommunicationCluster` within a physical network topology

Each ECU in the topology is modeled as an `EcuInstance`, which owns one or more `commController` - for sending frames to and receiving frames from the communication medium and `connector` - describing the bus interface and sending/receiving behavior. A `PhysicalChannel` references the `commConnector` of all participating ECUs to establish the topological connections within the cluster.

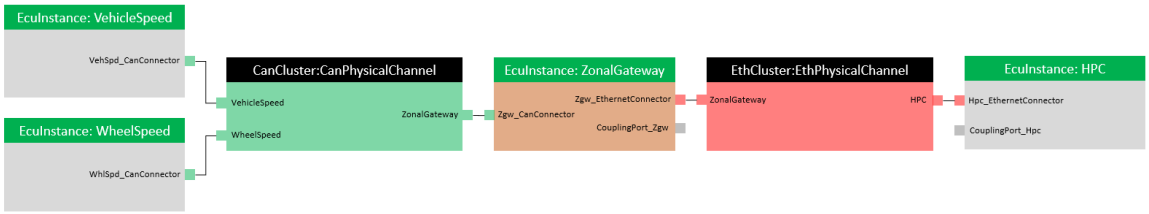


Figure 2.19: `EcuInstance` within a physical network topology

2.2.2.1 CAN

The WheelSpeed ECU and VehicleSpeed ECU are connected to the ZonalGateway ECU via a shared CAN. This network is modeled as a `CanCluster` - the CAN-specific specialization of `CommunicationCluster`. The `CanCluster` carries the common `CommunicationCluster`'s attributes:

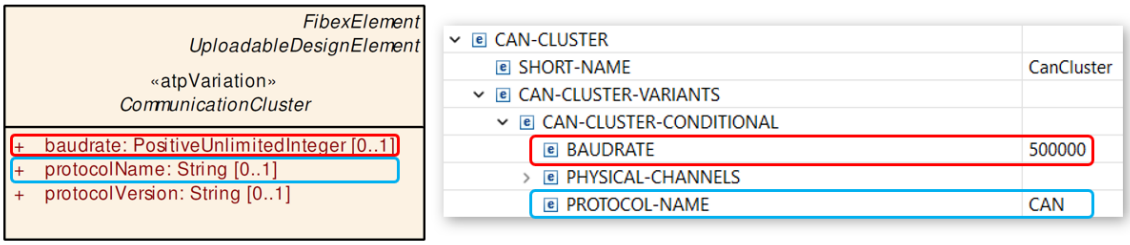


Figure 2.20: `CanCluster` configuration

Each ECU participating in the CAN cluster (WheelSpeed ECU, VehicleSpeed ECU, ZonalGateway ECU) holds:

- `CanCommunicationController` : is a specialization of the abstract `CommunicationController` class.

- `CanCommunicationConnector` : is a specialization of the abstract `CommunicationConnector` class.

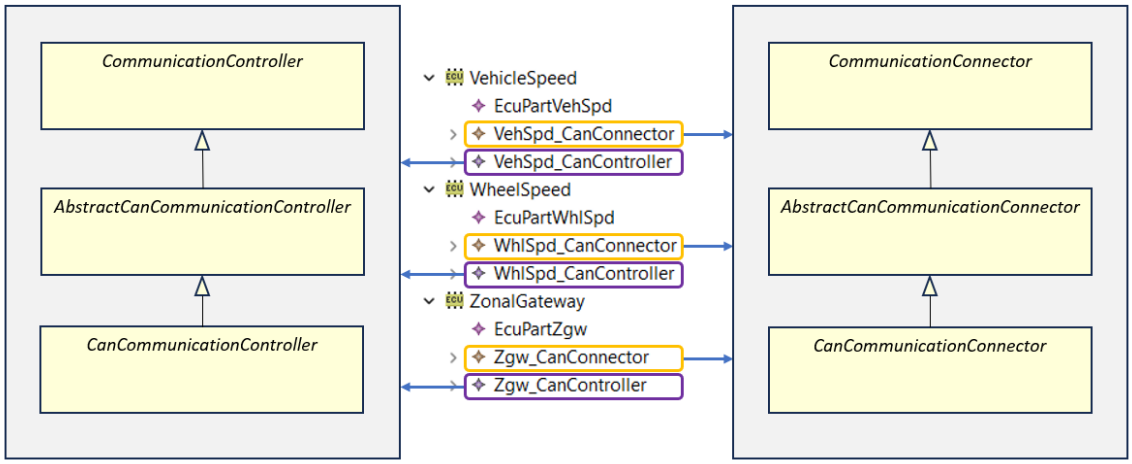


Figure 2.21: Specialized `CommunicationController` and `CanCommunicationConnector`

The `CanPhysicalChannel` represents the physical CAN bus wire. It references the `CanCommunicationConnector`s of WheelSpeed ECU, VehicleSpeed ECU and ZonalGateway ECU to establish which nodes participate on CAN channel:

PHYSICAL-CHANNELS	
CAN-PHYSICAL-CHANNEL	
SHORT-NAME	CanPhysicalChannel
COMM-CONNECTORS	
COMMUNICATION-CONNECTOR-REF-CONDITIONAL	
COMMUNICATION-CONNECTOR-REF	
DEST	CAN-COMMUNICATION-CONNECTOR
	/EcuInstances/WheelSpeed/WhlSpd_CanConnector
COMMUNICATION-CONNECTOR-REF-CONDITIONAL	
COMMUNICATION-CONNECTOR-REF	
DEST	CAN-COMMUNICATION-CONNECTOR
	/EcuInstances/VehicleSpeed/VehSpd_CanConnector
COMMUNICATION-CONNECTOR-REF-CONDITIONAL	
COMMUNICATION-CONNECTOR-REF	
DEST	CAN-COMMUNICATION-CONNECTOR
	/EcuInstances/ZonalGateway/Zgw_CanConnector
FRAME-TRIGGERINGS	
I-SIGNAL-TRIGGERINGS	
PDU-TRIGGERINGS	
PROTOCOL-NAME	CAN

Figure 2.22: `CanPhysicalChannel` references `CanCommunicationConnector`s of all connected ECUs

2.2.2.2 Ethernet

The ZonalGateway ECU and the HPC ECU are connected via Ethernet. This network is modeled as an EthernetCluster - the Ethernet-specific specialization of CommunicationCluster. The EthernetCluster carries the common CommunicationCluster's attributes:

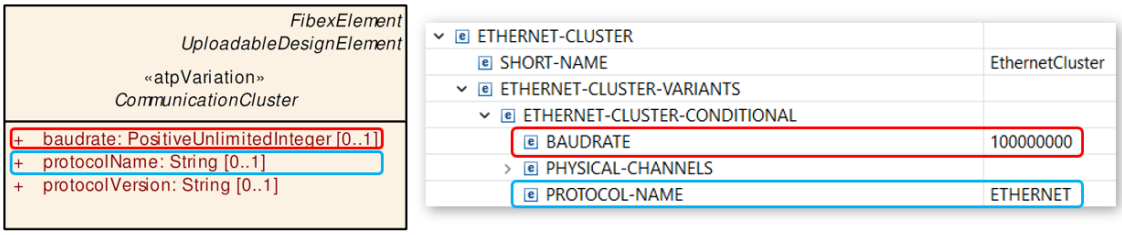


Figure 2.23: EthernetCluster configuration

Each ECU participating in the Ethernet cluster (ZonalGateway ECU, HPC ECU) holds:

- EthernetCommunicationController : is a specialization of the abstract CommunicationController class.
- EthernetCommunicationConnector : is a specialization of the abstract CommunicationConnector class.

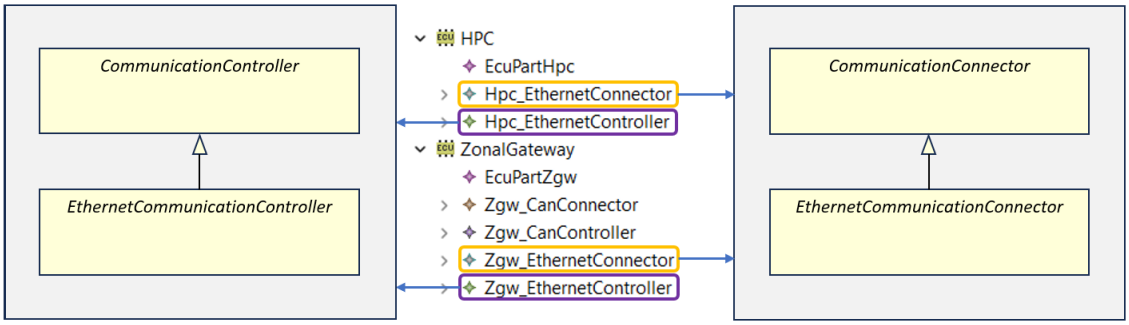


Figure 2.24: Specialized EthernetCommunicationController and EthernetCommunicationConnector

According to [TPS_SYST_02362] in [1], the value of the attribute baudrate in the context of an EthernetCluster has no meaning and shall be ignored. The actual communication speed is determined by the physicalLayerType attribute of the CouplingPort on each EthernetCommunicationConnector as described below:

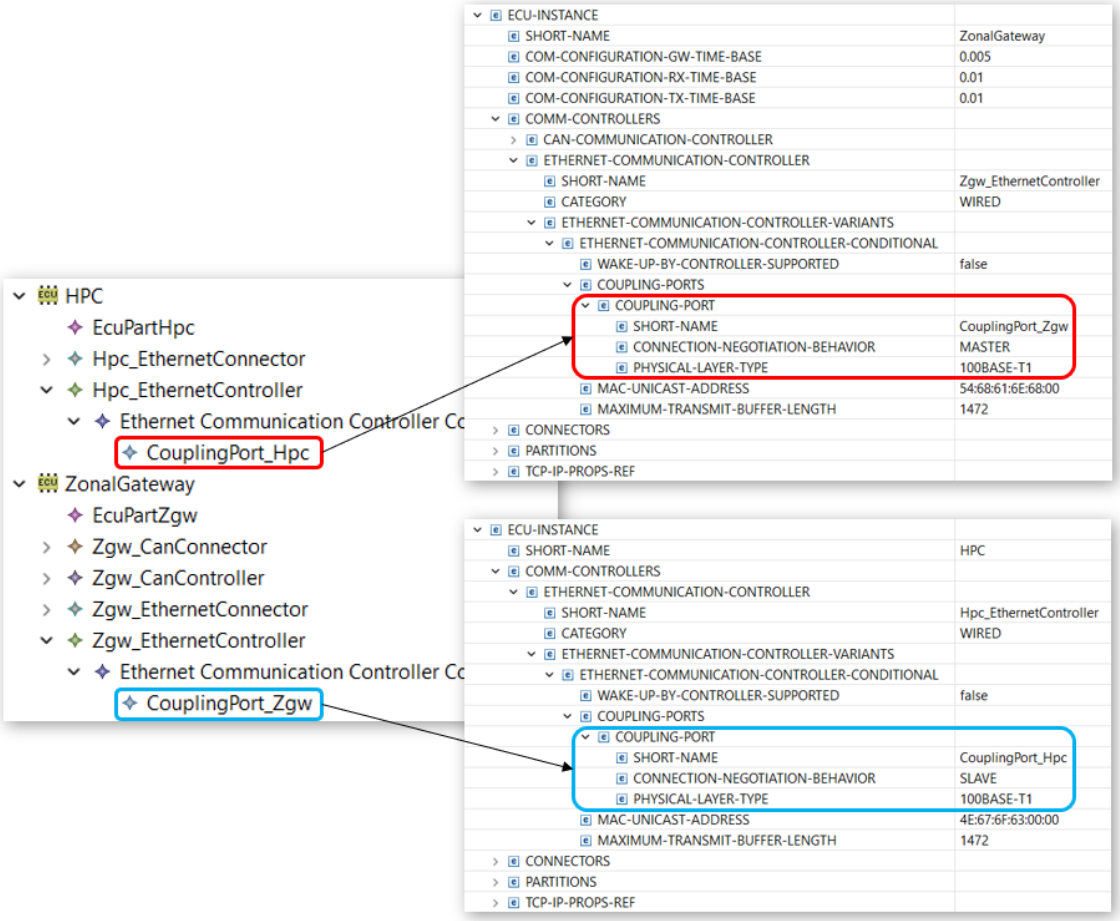


Figure 2.25: Each ECU on the Ethernet network holds an `EthernetCommunicationController` with a `CouplingPort`

To specify the communication speed `physicalLayerType` offers by `EthernetPhysicalLayerTypeEnum` dedicated values. In iteration two (R26-11) of the Workflow Example the value 100BASE-T1 is used [8].

2.2.2.3 Gateway

The ZonalGateway ECU serves as the `Gateway` element between the CAN Cluster and the Ethernet Cluster. It holds two separate `CommunicationController`s (Zgw_CanController on CAN, Zgw_EthernetController on Ethernet).

This system implements the Simple IPdu Gateway (reference to the section 8.2.1 in [1]):

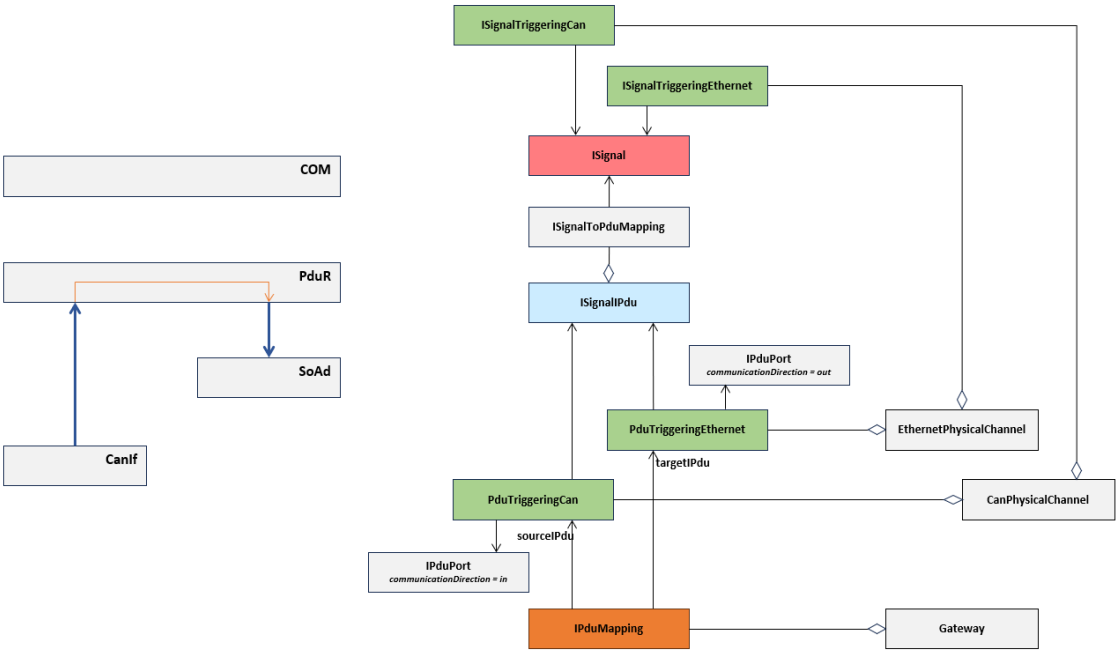


Figure 2.26: Simple IPdu Gateway CAN to Ethernet

In this illustration, both the source and target PduTriggerings of the IPduMapping reference to the same ISignalIPdu. The ISignalIPdu.iSignalToPduMapping references to ISignal having ISignalTriggering defined. However, this ISignalTriggering has no reference to an ISignalPort for ZonalGateway. Therefore, there is no interaction of the payload ISignal of the ISignalIPdu defined in the scope of ZonalGateway.

The ZonalGateway does not require application-level software for routing. The IPdu routing is performed by two IPduMappings defined in the Gateway element, covering both directions of data flow. Each IPduMapping pairs a sourceIPdu and a targetIPdu, both referencing to PduTriggerings on different PhysicalChannels.

Data	*Direction*	*Source*		*Target*	
		Source IPdu	*Source Physical Channel*	*Target IPdu*	*Target Physical Channel*
Wheel Speed	WhlSpd ECU → HPC ECU	PduTrig_Zgw_RoadWhlAgSpd_Rx	CanPhysicalChannel	PduTrig_Zgw_RoadWhlAgSpd_Tx	EthernetPhysicalChannel_VLAN10
Control Status	HPC ECU → VehSpd ECU	PduTrig_Zgw_CtrlSt_Rx	EthernetPhysicalChannel_VLAN10	PduTrig_Zgw_CtrlSt_Tx	CanPhysicalChannel

CAN
Ethernet

Figure 2.27: ZonalGateway is connected to CanPhysicalChannel and EthernetPhysicalChannel_VLAN10; and perform Pdu mapping between them

Mapping 1 - Wheel Speed CAN to Ethernet (WheelSpeed ECU to HPC ECU):

1. ZonalGateway ECU receives ISignalIPdu ISIPdu_Zgw_RoadWhlAgSpd (referenced by PduTriggering PduTrig_Zgw_RoadWhlAgSpd_Rx) from WheelSpeed ECU on the CAN network.
2. PduR applies IPduMapping and routes to EthernetPhysicalChannel_VLAN10 as PduTriggering PduTrig_Zgw_RoadWhlAgSpd_Tx.

3. ZonalGateway ECU sends `ISignalIPdu` ISIPdu_Zgw_RoadWhlAgSpd via UDP to `HPC` ECU on the Ethernet network.

Mapping 2 - Control Status Ethernet to CAN (`HPC` ECU to VehicleSpeed ECU):

1. ZonalGateway ECU receives `ISignalIPdu` ISIPdu_Zgw_CtrlSt (referenced by `PduTriggering` PduTrig_Zgw_CtrlSt_Rx) from `HPC` ECU on the Ethernet network.
2. PduR applies `IPduMapping` and routes to CanPhysicalChannel as `PduTriggering` PduTrig_Zgw_CtrlSt_Tx.
3. ZonalGateway ECU sends `ISignalIPdu` ISIPdu_Zgw_CtrlSt to VehicleSpeed ECU on the CAN network.

2.3 Differences to Previous Releases

This chapter spots the differences to previous releases of the Workflow Example. It is not intended to repeat the content in detail. The reader can select the previous version of the document in the AUTOSAR repository.

In the first iteration (R25-11) of the Workflow Example

- only two ECUs ("WheelSpeed" and "VehicleSpeed") are connected via CAN
- two architectural approaches - top-down and bottom-up - are described to motivate different aspects, illustrated in the `SystemTemplate` [1]

3 Related Documentation

[1] System Template
AUTOSAR_CP_TPS_SystemTemplate

[2] Generic Structure Template
AUTOSAR_FO_TPS_GenericStructureTemplate

[3] Software Component Template
AUTOSAR_CP_TPS_SoftwareComponentTemplate

[4] Virtual Functional Bus
AUTOSAR_CP_TR_VFB

[5] Layered Software Architecture
AUTOSAR_CP_EXP_LayeredSoftwareArchitecture

[6] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[7] Specification of RTE Software
AUTOSAR_CP_SWS_RTE

[8] IEEE 802.1Q-2022 - IEEE Standard for Local and Metropolitan Area Network - Bridges and Bridged Networks
https://ieeexplore.ieee.org/

[9] AUTOSAR Webpage – Classic Platform
https://www.autosar.org/standards/classic-platform

A Delivered artefacts

The delivery of the second iteration (R26-11) of the Workflow Example (*WorkflowExample.zip*) includes the following artefacts:

These are:

- `ECU_SYSTEM_DESCRIPTION` of the Workflow Example project containing all relevant ARXML files regarding to the infrastructure, software components and system configuration. Used and provided by the OEM.
- `SYSTEM_EXTRACT` of ECU 1 for the WheelSpeed composition (*WheelSpeed__SystemExtract.arxml*), used by the supplier of ECU 1 for further processing.
- `SYSTEM_EXTRACT` of ECU 2 for the VehicleSpeed composition (*VehicleSpeed__SystemExtract.arxml*), used by the supplier of ECU 2 for further processing.
- `SYSTEM_EXTRACT` of ECU 3 for the `HPC` composition (*Hpc__SystemExtract.arxml*), used by the supplier of ECU 3 for further processing.
- `SYSTEM_EXTRACT` of ECU 4 for the ZonalGateway (*ZonalGateway__SystemExtract.arxml*), used by the supplier of ECU 4 for further processing.
- *AUTOSAR_CP_TR_WorkflowExample.pdf*, documentation of the Workflow Example, this file.

Instruction for user of delivered artefacts!

1. Download the *WorkflowExample.zip* from the AUTOSAR Webpage - Classic Platform [9].
2. Unzip *WorkflowExample.zip*, five folders will be created:
   - EcuSystemDescription,
   - Ecu1SystemExtract,
   - Ecu2SystemExtract,
   - Ecu3SystemExtract and
   - Ecu4SystemExtract.
3. Depending on your role (`OEM`, `supplier`) load the corresponding folder in your AUTOSAR tooling and start working. As an `OEM` work based on the EcuSystemDescription, as a `supplier` use the System Extracts of ECU1, ECU2, ECU3 or ECU4.
4. As `supplier` continue as illustrated in figure 2.4 and complete the sub-system.
5. As `OEM` integrate the provided ECUs as illustrated in figure 2.7.

6. Job done.

B Mentioned Class Tables

For the sake of completeness, this chapter contains a set of class tables representing meta-classes mentioned in the context of this document but which are not contained directly in the scope of describing specific meta-model semantics.

Class	***ARElement*** (abstract)			
Note	An element that can be defined stand-alone, i.e. without being part of another element (except for packages of course).			
Base	*ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*AbstractMessageToSecureChannelMapping*, AclObjectSet, AclOperation, AclPermission, AclRole, Alias NameSet, ApplicabilityInfoSet, ApplicationPartition, *AutosarDataType*, *BaseType*, BlueprintMappingSet, BswEntryRelationshipSet, BswModuleDescription, BswModuleEntry, BuildActionManifest, Calibration ParameterValueSet, ClientIdDefinitionSet, Collection, CompuMethod, ConsistencyNeedsBlueprintSet, ConstantSpecification, ConstantSpecificationMappingSet, CpSoftwareCluster, CpSoftwareClusterBinary ManifestDescriptor, CpSoftwareClusterMappingSet, CpSoftwareClusterResourcePool, CryptoElliptic CurveProps, CryptoServiceCertificate, CryptoServiceKey, CryptoServicePrimitive, CryptoServiceQueue, CryptoSignatureScheme, DataConstr, DataTransformationSet, DataTypeMappingSet, DdsCpConfig, *DiagnosticCommonElement*, DiagnosticConnection, DiagnosticContributionSet, DltArgumentPropsSet, DltContext, DltEcu, Documentation, E2EProfileCompatibilityProps, EcucDefinitionCollection, Ecuc DestinationUriDefSet, EcucModuleConfigurationValues, EcucModuleDef, EcucValueCollection, EthIp Props, EthTcpIpIcmpProps, EthTcpIpProps, EvaluatedVariantSet, FMFeature, FMFeatureMap, FMFeatureModel, FMFeatureSelectionSet, FirewallRule, FlatMap, GeneralPurposeConnection, Hw Category, HwElement, HwType, *IEEE1722TpConnection*, IPSecConfigProps, IPv6ExtHeaderFilterSet, *IdsCommonElement*, IdsDesign, *Implementation*, ImpositionTimeDefinitionGroup, InterpolationRoutine MappingSet, J1939ControllerApplication, KeywordSet, LifeCycleInfoSet, LifeCycleStateDefinitionGroup, LogAndTraceMessageCollectionSet, MacSecGlobalKayProps, MacSecParticipantSet, McFunction, Mc Group, ModeDeclarationGroup, ModeDeclarationMappingSet, OsTaskProxy, PhysicalDimension, PhysicalDimensionMappingSet, *PortInterface*, PortInterfaceMappingSet, PortPrototypeBlueprint, Post BuildVariantCriterion, PostBuildVariantCriterionValueSet, PredefinedVariant, RapidPrototypingScenario, SdgDef, *SecureComProps*, *SecurityConnectivityAssociation*, SignalServiceTranslationPropsSet, Someip SdClientEventGroupTimingConfig, SomeipSdClientServiceInstanceConfig, SomeipSdServerEventGroup TimingConfig, SomeipSdServerServiceInstanceConfig, SwAddrMethod, SwAxisType, SwComponent MappingConstraints, *SwComponentType*, SwRecordLayout, SwSystemconst, SwSystemconstantValue Set, SwcBswMapping, System, SystemComSpecDefinitionSet, SystemSignal, SystemSignalGroup, TDCpSoftwareClusterMappingSet, TcpOptionFilterSet, *TimingExtension*, TlsConnectionGroup, TlvData IdDefinitionSet, TransformationPropsSet, Unit, UnitGroup, *UploadablePackageElement*, ViewMapSet			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table B.1: ARElement

Class	**ARPackage**			
Note	AUTOSAR package, allowing to create top level packages to structure the contained ARElements. ARPackages are open sets. This means that in a file based description system multiple files can be used to partially describe the contents of a package. This is an extended version of MSR's SW-SYSTEM.			
Base	*ARObject*, *AtpBlueprint*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ARPackage.arPackage, AUTOSAR.arPackage			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**ARPackage**			
arPackage	ARPackage	*	aggr	This represents a sub package within an ARPackage, thus allowing for an unlimited package hierarchy. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=arPackage, arPackage.shortName, ar Package.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=30
element	PackageableElement	*	aggr	Elements that are part of this package **Stereotypes:** atpVariation **Tags:** atp.Splitkey=element, element.shortName, element.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime xml.sequenceOffset=20
referenceBase	ReferenceBase	*	aggr	This denotes the reference bases for the package. This is the basis for all relative references within the package. The base needs to be selected according to the base attribute within the references. **Tags:** atp.Splitkey=referenceBase, referenceBase.shortLabel xml.sequenceOffset=10

Table B.2: ARPackage

Class	**ApplicationSwComponentType**			
Note	The `ApplicationSwComponentType` is used to represent the application software. **Tags:** atp.recommendedPackage=SwComponentTypes			
Base	*ARElement*, *ARObject*, *AtomicSwComponentType*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *Collectable Element*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table B.3: ApplicationSwComponentType

Class	**AssemblySwConnector**			
Note	`AssemblySwConnector`s are exclusively used to connect `SwComponentPrototype`s in the context of a `CompositionSwComponentType`.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SwConnector*			
Aggregated by	*AtpClassifier*.atpFeature, CompositionSwComponentType.connector			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
provider	AbstractProvidedPort Prototype	0..1	iref	Instance of providing port. **Tags:** atp.Splitkey=provider, provider.contextComponent, provider.targetPPort **InstanceRef implemented by:** PPortInComposition InstanceRef

▽

△

Class	**AssemblySwConnector**			
requester	AbstractRequiredPort Prototype	0..1	iref	Instance of requiring port. **Tags:** atp.Splitkey=requester, requester.context Component, requester.targetRPort **InstanceRef implemented by:** RPortInComposition InstanceRef

Table B.4: AssemblySwConnector

Class	«atpVariation» **CanCluster**			
Note	CAN bus specific cluster attributes. **Tags:** atp.recommendedPackage=CommunicationClusters			
Base	*ARElement*, *ARObject*, *AbstractCanCluster*, *CollectableElement*, *CommunicationCluster*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table B.5: CanCluster

Class	**CanCommunicationConnector**			
Note	CAN bus specific communication connector attributes.			
Base	*ARObject*, *AbstractCanCommunicationConnector*, *CommunicationConnector*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcuInstance.connector, MachineDesign.communicationConnector			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
pncWakeupCanId	PositiveInteger	0..1	attr	CAN Identifier used to configure the CAN Transceiver for partial network wakeup.
pncWakeupCanIdExtended	Boolean	0..1	attr	Defines whether pncWakeupCanId and pncWakeupCanId Mask shall be interpreted as extended or standard CAN ID.
pncWakeupCanIdMask	PositiveInteger	0..1	attr	Bit mask for CAN Identifier used to configure the CAN Transceiver for partial network wakeup.
pncWakeup DataMask	PositiveUnlimitedInteger	0..1	attr	Bit mask for CAN Payload used to configure the CAN Transceiver for partial network wakeup.
pncWakeupDlc	PositiveInteger	0..1	attr	Data Length of the remote data frame used to configure the CAN Transceiver for partial network wakeup in Bytes.

Table B.6: CanCommunicationConnector

Class	«atpVariation» **CanCommunicationController**			
Note	CAN bus specific communication port attributes.			
Base	*ARObject*, *AbstractCanCommunicationController*, *CommunicationController*, *Identifiable*, *Multilanguage Referrable*, *Referrable*			
Aggregated by	EcuInstance.commController, MachineDesign.communicationController			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table B.7: CanCommunicationController

Class	**CanPhysicalChannel**			
Note	CAN bus specific physical channel attributes.			
Base	*ARObject*, *AbstractCanPhysicalChannel*, *Identifiable*, *MultilanguageReferrable*, *PhysicalChannel*, *Referrable*			
Aggregated by	*CommunicationCluster*.physicalChannel			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table B.8: CanPhysicalChannel

Class	«atpVariation» ***CommunicationCluster*** (abstract)			
Note	The CommunicationCluster is the main element to describe the topological connection of communicating ECUs. A cluster describes the ensemble of ECUs, which are linked by a communication medium of arbitrary topology (bus, star, ring, ...). The nodes within the cluster share the same communication protocol, which may be event-triggered, time-triggered or a combination of both. A CommunicationCluster aggregates one or more physical channels. **Tags:** vh.latestBindingTime=postBuild			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Subclasses	*AbstractCanCluster*, EthernetCluster, FlexrayCluster, LinCluster, UserDefinedCluster			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
baudrate	PositiveUnlimitedInteger	0..1	attr	Channels speed in bits/s.
comMLight Timeout	TimeValue	0..1	attr	Defines the timeout (in seconds) after COMM_FULL_ COMMUNICATION sub-state COMM_FULL_COM_ READY_SLEEP is left in the ComM channel state machine.
physical Channel	PhysicalChannel	*	aggr	This relationship defines which channel element belongs to which cluster. A channel shall be assigned to exactly one cluster, whereas a cluster may have one or more channels. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=physicalChannel, physicalChannel.short Name, physicalChannel.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime
protocolName	String	0..1	attr	The name of the protocol used.
protocolVersion	String	0..1	attr	The version of the protocol used.

Table B.9: CommunicationCluster

Class	***CommunicationConnector*** (abstract)
Note	The connection between the referencing ECU and the referenced channel via the referenced controller. Connectors are used to describe the bus interfaces of the ECUs and to specify the sending/receiving behavior. Each CommunicationConnector has a reference to exactly one communicationController. Note: Several CommunicationConnectors can be assigned to one PhysicalChannel in the scope of one ECU Instance.
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*
Subclasses	*AbstractCanCommunicationConnector*, EthernetCommunicationConnector, FlexrayCommunication Connector, LinCommunicationConnector, UserDefinedCommunicationConnector
Aggregated by	EcuInstance.connector, MachineDesign.communicationConnector

▽

△

Class	*CommunicationConnector* (abstract)			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
commController	Communication Controller	0..1	ref	Reference to the communication controller. The CommunicationConnector and referenced CommunicationController shall be aggregated by the same ECUInstance. The communicationController can be referenced by several CommunicationConnector elements. This is important for the FlexRay Bus. FlexRay communicates via two physical channels. But only one controller in an ECU is responsible for both channels. Thus, two connectors (for channel A and for channel B) shall reference to the same controller. **Tags:** atp.Splitkey=commController
createEcu WakeupSource	Boolean	0..1	attr	If this parameter is available and set to true then a channel wakeup source shall be created for the Physical Channel referencing this CommunicationConnector.
dynamicPncTo Channel Mapping Enabled	Boolean	0..1	attr	Defines if this EcuInstance shall implement the dynamic PNC-to-channel-mapping functionality on this CommunicationConnector and its respective Physical Channel. **Tags:** atp.Status=draft This Attribute is only used by the AUTOSAR Classic Platform.
ecuCommPort Instance	CommConnectorPort	*	aggr	An ECUs reception or send ports. atpVariation: If signals/PDUs/frames are variable, the corresponding ports shall be variable, too. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=ecuCommPortInstance, ecuCommPort Instance.shortName, ecuCommPortInstance.variation Point.shortLabel vh.latestBindingTime=postBuild This Attribute is only used by the AUTOSAR Classic Platform.
explicitWakeup Channel	PhysicalChannel	*	ref	Defines a restriction which PhysicalChannels shall be woken up if this CommunicationConnector is the wakeup source. If not defined, then no restriction applies. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=explicitWakeupChannel, explicitWakeup Channel.physicalChannel, explicitWakeup Channel.variationPoint.shortLabel vh.latestBindingTime=postBuild
explicitWakeup Pnc	PncMappingIdent	*	ref	Defines a restriction which PNCs shall be woken up if this CommunicationConnector is the wakeup source. If not defined, then no restriction applies. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=explicitWakeupPnc, explicitWakeupPnc.pnc MappingIdent, explicitWakeupPnc.variationPoint.short Label vh.latestBindingTime=postBuild
nmVariant	NmVariantEnum	0..1	attr	Defines the functionality of Network Management.
pncFilterArray Mask (ordered)	PositiveInteger	*	attr	Bit mask for NM-Pdu Payload used to configure the NM filter mask for the Network Management.

▽

△

Class	***CommunicationConnector*** (abstract)			
pncGateway Type	PncGatewayTypeEnum	0..1	attr	Defines if this EcuInstance shall implement the Pnc Gateway functionality on this CommunicationConnector and its respective PhysicalChannel. Several Ecu Instances on the same PhysicalChannel can have the PncGateway functionality enabled, but only one of them shall have the pncGatewayType "active". This Attribute is only used by the AUTOSAR Classic Platform.

Table B.10: CommunicationConnector

Class	«atpVariation» ***CommunicationController*** (abstract)			
Note	The communication controller is a dedicated hardware device by means of which hosts are sending frames to and receiving frames from the communication medium. **Tags:** vh.latestBindingTime=postBuild			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*AbstractCanCommunicationController*, EthernetCommunicationController, FlexrayCommunication Controller, *LinCommunicationController*, UserDefinedCommunicationController			
Aggregated by	EcuInstance.commController, MachineDesign.communicationController			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
wakeUpBy Controller Supported	Boolean	0..1	attr	Defines whether the ECU shall be woken up by this CommunicationController. TRUE: wake up is possible FALSE: wake up is not supported Note: If wakeUpByControllerSupported is set to TRUE the feature shall be supported by both hardware and basic software.

Table B.11: CommunicationController

Class	**CompositionSwComponentType**			
Note	A `CompositionSwComponentType` aggregates `SwComponentPrototype`s (that in turn are typed by `SwComponentType`)s as well as `SwConnector`s for primarily connecting `SwComponentPrototype`s among each others and towards the surface of the `CompositionSwComponentType`. By this means, a hierarchical structures of software-components can be created. **Tags:** atp.recommendedPackage=SwComponentTypes			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**CompositionSwComponentType**			
component	SwComponent Prototype	*	aggr	The instantiated components that are part of this composition. The aggregation of `SwComponentPrototype` is subject to variability with the purpose to support the conditional existence of a `SwComponentPrototype`. Please be aware: if the conditional existence of `SwComponentPrototype`s is resolved post-build, the deselected `SwComponentPrototype`s are still contained in the ECUs build but the instances are inactive in that they are not scheduled by the RTE. The use case for having 0 components owned by the `CompositionSwComponentType` could be to deliver an empty `CompositionSwComponentType` to e.g. a supplier for filling the internal structure. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=component, component.shortName, component.variationPoint.shortLabel vh.latestBindingTime=postBuild
connector	SwConnector	*	aggr	`SwConnector`s have the principal ability to establish a connection among `PortPrototype`s. They can have many roles in the context of a `CompositionSwComponentType`. Details are refined by subclasses. The aggregation of `SwConnector`s is subject to variability with the purpose to support variant data flow. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=connector, connector.shortName, connector.variationPoint.shortLabel vh.latestBindingTime=postBuild
constantValue Mapping	ConstantSpecification MappingSet	*	ref	Reference to the `ConstantSpecificationMapping` to be applied for initValues of `PPortComSpec`s and `RPortComSpec`. **Tags:** atp.Splitkey=constantValueMapping
dataType Mapping	DataTypeMappingSet	*	ref	Reference to the `DataTypeMappingSet` to be applied for the used `ApplicationDataType`s in `PortInterface`s. Background: when developing subsystems it may happen that `ApplicationDataType`s are used on the surface of `CompositionSwComponentType`s. In this case it would be reasonable to be able to also provide the intended mapping to the `ImplementationDataType`s. However, this mapping shall be informal and not technically binding for the implementors mainly because the RTE generator is not concerned about the `CompositionSwComponentType`s. Rationale: if the mapping of `ApplicationDataType`s on the delegated and inner `PortPrototype` matches then the mapping to `ImplementationDataType`s is not impacting compatibility. **Tags:** atp.Splitkey=dataTypeMapping

▽

△

Class	**CompositionSwComponentType**			
instantiation RTEEventProps	InstantiationRTEEvent Props	*	aggr	This allows to define instantiation specific properties for RTE Events, in particular for instance specific scheduling. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=instantiationRTEEventProps, instantiation RTEEventProps.shortLabel, instantiationRTEEvent Props.variationPoint.shortLabel vh.latestBindingTime=codeGenerationTime This Attribute is only used by the AUTOSAR Classic Platform.
physical Dimension Mapping	PhysicalDimension MappingSet	0..1	ref	This reference identifies the `PhysicalDimensionMappingSet` that is applicable in the context of the enclosing `CompositionSwComponentType`. The `PhysicalDimensionMapping`s contained in the `PhysicalDimensionMappingSet` shall be taken into account for the assessment of the compatibility of `PhysicalDimension`s in the context of creation of a `PortInterfaceMapping` in the scope of the `CompositionSwComponentType`. **Tags:** atp.Splitkey=physicalDimensionMapping

Table B.12: CompositionSwComponentType

Class	**CouplingPort**			
Note	A CouplingPort is used to connect a CouplingElement with an EcuInstance or two CouplingElements with each other via a CouplingPortConnection. Optionally, the CouplingPort may also have a reference to a macMulticastGroup and a defaultVLAN.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	CouplingElement.couplingPort, EthernetCommunicationController.couplingPort			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
connection Negotiation Behavior	EthernetConnection NegotiationEnum	0..1	attr	Specifies the connection negotiation of the CouplingPort.
couplingPort Details	CouplingPortDetails	0..1	aggr	Defines more details of a CouplingPort in case a more specific configuration is required. **Tags:** atp.Splitkey=couplingPortDetails
couplingPort Role	CouplingPortRoleEnum	0..1	attr	Defines the role this CouplingPort takes in the context of the CouplingElement.
defaultVlan	EthernetPhysical Channel	0..1	ref	The vLanIdentifier of the referenced VLAN is the Default-PVID (port VLAN ID). A Port VLAN ID is a default VLAN ID that is assigned to an access CouplingPort to designate the VLAN segment to which this port is connected. Also, if a CouplingPort has not been configured with any VLAN memberships, the virtual switch's Port VLAN ID (pvid) becomes the default VLAN ID for the ports connection. This identifier/tag is added for incoming untagged messages at the port (ingress tagging). For outgoing messages with this identifier, the tag is removed at the port (egress untagging, depending on the Vlan Membership.sendActivity). **Tags:** atp.Splitkey=defaultVlan

▽

△

Class	**CouplingPort**			
macAddress VlanAssignment	MacAddressVlan Membership	*	aggr	Statically defines the assignment of MAC-Multicast-Addresses, optionally together with VLANs, to this CouplingPort. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=macAddressVlanAssignment, macAddress VlanAssignment.shortName, macAddressVlan Assignment.variationPoint.shortLabel vh.latestBindingTime=postBuild
macLayerType	EthernetMacLayerType Enum	0..1	attr	Specifies the mac layer type of the CouplingPort.
macSecProps	MacSecProps	*	aggr	Properties to configure MACsec (Media access control security) and the MKA (MACsec Key Agreement) for the CouplingPort (PHY). **Tags:** atp.Splitkey=macSecProps atp.Status=candidate
physicalLayer Type	EthernetPhysicalLayer TypeEnum	0..1	attr	Specifies the physical layer type of the CouplingPort.
plcaProps	PlcaProps	0..1	aggr	Optional properties for configuration of PLCA (Physical Layer Collision Avoidance) in case 10-BASE-T1S Ethernet is used and PLCA is enabled on the Coupling Port (PHY). **Tags:** atp.Splitkey=plcaProps
pncMapping	PncMappingIdent	*	ref	Reference to the partial networks this CouplingPort participates in. **Tags:** atp.Splitkey=pncMapping
receiveActivity	EthernetSwitchVlan IngressTagEnum	0..1	attr	Defines the handling of frames at the ingress port.
vlan Membership	VlanMembership	*	aggr	Messages of VLANs that are defined here can be communicated via the CouplingPort. **Tags:** atp.Splitkey=vlanMembership
wakeupSleep OnDataline Config	EthernetWakeupSleep OnDatalineConfig	0..1	ref	Optional reference to EthernetWakeupSleepOnDataline Config. **Tags:** atp.Splitkey=wakeupSleepOnDatalineConfig

Table B.13: CouplingPort

Class	***DataMapping*** (abstract)			
Note	Mapping of port elements (data elements and parameters) to frames and signals.			
Base	*ARObject*			
Subclasses	ClientServerToSignalMapping, SenderReceiverToSignalGroupMapping, SenderReceiverToSignal Mapping, TriggerToSignalMapping			
Aggregated by	SystemMapping.dataMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
introduction	DocumentationBlock	0..1	aggr	This represents introductory documentation about the data mapping. **Tags:** atp.Splitkey=introduction

Table B.14: DataMapping

Class	**DelegationSwConnector**			
Note	A delegation connector delegates one inner `PortPrototype` (a port of a component that is used inside the composition) to a outer `PortPrototype` of compatible type that belongs directly to the composition (a port that is owned by the composition).			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SwConnector*			
Aggregated by	*AtpClassifier*.atpFeature, CompositionSwComponentType.connector			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
innerPort	PortPrototype	0..1	iref	The port that belongs to the ComponentPrototype in the composition **Tags:** atp.Splitkey=innerPort, innerPort.context Component, innerPort.targetPPort, innerPort.targetRPort **InstanceRef implemented by:** PortInCompositionType InstanceRef
outerPort	PortPrototype	0..1	ref	The port that is located on the outside of the Composition Type **Tags:** atp.Splitkey=outerPort

Table B.15: DelegationSwConnector

Class	**EcuAbstractionSwComponentType**			
Note	The ECUAbstraction is a special AtomicSwComponentType that resides between a software-component that wants to access ECU periphery and the Microcontroller Abstraction. The EcuAbstractionSw ComponentType introduces the possibility to link from the software representation to its hardware description provided by the ECU Resource Template. **Tags:** atp.recommendedPackage=SwComponentTypes			
Base	*ARElement*, *ARObject*, *AtomicSwComponentType*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *Collectable Element*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
hardware Element	HwDescriptionEntity	*	ref	Reference from the EcuAbstractionComponentType to the description of the used HwElements. **Tags:** atp.Splitkey=hardwareElement

Table B.16: EcuAbstractionSwComponentType

Class	**EcuInstance**			
Note	ECUInstances are used to define the ECUs used in the topology. The type of the ECU is defined by a reference to an ECU specified with the ECU resource description. **Tags:** atp.recommendedPackage=EcuInstances			
Base	*ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	EcuInstance			
associatedComIPduGroup	ISignalIPduGroup	*	ref	With this reference it is possible to identify which ISignal IPduGroups are applicable for which Communication Connector/ ECU. Only top level ISignalIPduGroups shall be referenced by an EcuInstance. If an ISignalIPduGroup contains other ISignalIPduGroups than these contained ISignalIPdu Groups shall not be referenced by the EcuInstance. Contained ISignalIPduGroups are associated to an Ecu Instance via the top level ISignalIPduGroup. **Tags:** atp.Splitkey=associatedComIPduGroup This Attribute is only used by the AUTOSAR Classic Platform.
associated Consumed Provided ServiceInstance Group	ConsumedProvided ServiceInstanceGroup	*	ref	With this reference it is possible to identify which ConsumedProvidedServiceInstanceGroups are applicable for which ECUInstance. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=associatedConsumedProvidedService InstanceGroup, associatedConsumedProvidedService InstanceGroup.consumedProvidedServiceInstanceGroup, associatedConsumedProvidedServiceInstance Group.variationPoint.shortLabel vh.latestBindingTime=postBuild
associatedPdurIPduGroup	PdurIPduGroup	*	ref	With this reference it is possible to identify which PduR IPdu Groups are applicable for which Communication Connector/ ECU. **Tags:** atp.Splitkey=associatedPdurIPduGroup
channel Synchronous Wakeup	Boolean	0..1	attr	If this parameter is available and set to true, then all available channels will be woken up as soon as at least one channel wakeup occurs. If PNCs are configured, then all PNCs will be requested upon a channel wakeup.
clientIdRange	ClientIdRange	0..1	aggr	Restriction of the Client Identifier for this Ecu to an allowed range of numerical values. The Client Identifier of the transaction handle is generated by the client RTE for inter-Ecu Client/Server communication. **Tags:** atp.Splitkey=clientIdRange
com Configuration GwTimeBase	TimeValue	0..1	attr	The period between successive calls to Com_Main FunctionRouteSignals of the AUTOSAR COM module in seconds. This Attribute is only used by the AUTOSAR Classic Platform.
com ConfigurationRx TimeBase	TimeValue	0..1	attr	The period between successive calls to Com_Main FunctionRx of the AUTOSAR COM module in seconds. This Attribute is only used by the AUTOSAR Classic Platform.
com ConfigurationTx TimeBase	TimeValue	0..1	attr	The period between successive calls to Com_Main FunctionTx of the AUTOSAR COM module in seconds. This Attribute is only used by the AUTOSAR Classic Platform.
comEnable MDTForCyclic Transmission	Boolean	0..1	attr	Enables for the Com module of this EcuInstance the minimum delay time monitoring for cyclic and repeated transmissions (TransmissionModeTiming has cyclic Timing assigned or eventControlledTiming with numberOf Repetitions > 0). This Attribute is only used by the AUTOSAR Classic Platform.

▽

△

Class	EcuInstance			
commController	Communication Controller	*	aggr	CommunicationControllers of the ECU. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=commController, commController.shortName, commController.variationPoint.shortLabel vh.latestBindingTime=postBuild
connector	Communication Connector	*	aggr	All channels controlled by a single controller. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=connector, connector.shortName, connector.variationPoint.shortLabel vh.latestBindingTime=postBuild
dltConfig	DltConfig	0..1	aggr	Describes the Dlt configuration on this EcuInstance. **Tags:** atp.Splitkey=dltConfig This Attribute is only used by the AUTOSAR Classic Platform.
doIpConfig	DoIpConfig	0..1	aggr	DoIp configuration on this EcuInstance. **Tags:** atp.Splitkey=doIpConfig atp.Status=draft This Attribute is only used by the AUTOSAR Classic Platform.
ecuTaskProxy	OsTaskProxy	*	ref	Reference to OsTaskProxies assigned to the Ecu Instance. **Tags:** atp.Splitkey=ecuTaskProxy This Attribute is only used by the AUTOSAR Classic Platform.
ethSwitchPort Group Derivation	Boolean	0..1	attr	Defines whether the derivation of SwitchPortGroups based on VLAN and/or CouplingPort.pncMapping shall be performed for this EcuInstance. If not defined the derivation shall not be done. This Attribute is only used by the AUTOSAR Classic Platform.
firewallRule	StateDependentFirewall	*	ref	Firewall rules defined in the context of an EcuInstance. **Tags:** atp.Splitkey=firewallRule
j1939Node	J1939Node	*	aggr	Optional collection of J1939Nodes defined on this Ecu Instance. **Tags:** atp.Splitkey=j1939Node, j1939Node.shortName This Attribute is only used by the AUTOSAR Classic Platform.
partition	EcuPartition	*	aggr	Optional definition of Partitions within an Ecu. **Tags:** atp.Splitkey=partition, partition.shortName This Attribute is only used by the AUTOSAR Classic Platform.
pncNmRequest	Boolean	0..1	attr	Defines if this EcuInstance shall request Nm on all its PhysicalChannels which have Nm variant set to FULL each time a PNC is requested.
pncPrepare SleepTimer	TimeValue	0..1	attr	Time in seconds the PNC state machine shall wait in PNC_PREPARE_SLEEP.
pnc Synchronous Wakeup	Boolean	0..1	attr	If this parameter is available and set to true then all available PNCs will be woken up as soon as a channel wakeup occurs. This is ensured by adding all PNCs to all channel wakeup sources during upstream mapping.

▽

△

Class	**EcuInstance**			
pnResetTime	TimeValue	0..1	attr	Specifies the runtime of the reset timer in seconds. This reset time is valid for the reset of PN requests in the EIRA and in the ERA.
sleepMode Supported	Boolean	0..1	attr	Specifies whether the ECU instance may be put to a "low power mode" • true: sleep mode is supported • false: sleep mode is not supported Note: This flag may only be set to "true" if the feature is supported by both hardware and basic software. This Attribute is only used by the AUTOSAR Classic Platform.
tcpIpIcmpProps	EthTcpIpIcmpProps	0..1	ref	EcuInstance specific ICMP (Internet Control Message Protocol) attributes **Tags:** atp.Splitkey=tcpIpIcmpProps This Attribute is only used by the AUTOSAR Classic Platform.
tcpIpProps	EthTcpIpProps	0..1	ref	EcuInstance specific TcpIp Stack attributes. **Tags:** atp.Splitkey=tcpIpProps This Attribute is only used by the AUTOSAR Classic Platform.
wakeUpOver BusSupported	Boolean	0..1	attr	Driver support for wakeup over Bus. This Attribute is only used by the AUTOSAR Classic Platform.

Table B.17: EcuInstance

Class	«atpVariation» **EthernetCluster**			
Note	Ethernet-specific cluster attributes. **Tags:** atp.recommendedPackage=CommunicationClusters			
Base	*ARElement*, *ARObject*, *CollectableElement*, *CommunicationCluster*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *Uploadable PackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
couplingPort Connection	CouplingPort Connection	*	aggr	Specification of connections between CouplingElements and EcuInstances. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=couplingPortConnection, couplingPort Connection.variationPoint.shortLabel vh.latestBindingTime=postBuild
couplingPort StartupActive Time	TimeValue	0..1	attr	The attribute specifies the time in second a coupling port is switched on to enable the host ECU (ECU that maintains an Ethernet switch) to listen to the network for potential network management requests.
couplingPort SwitchoffDelay	TimeValue	0..1	attr	Switch off delay for CouplingPorts in seconds. It denotes the delay of switching off couplingPorts after the request to switch off a couplingPort was issued. (e.g. switch off of Ethernet switch ports).

▽

△

Class	«atpVariation» **EthernetCluster**			
macMulticast Group	MacMulticastGroup	*	aggr	MacMulticastGroup that is defined for the Subnet (EthernetCluster). **Tags:** atp.Splitkey=macMulticastGroup, macMulticast Group.shortName

Table B.18: EthernetCluster

Class	**EthernetCommunicationConnector**			
Note	Ethernet specific attributes to the CommunicationConnector.			
Base	*ARObject*, *CommunicationConnector*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcuInstance.connector, MachineDesign.communicationConnector			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ethIpProps	EthIpProps	0..1	ref	EcuInstance specific IP attributes. **Tags:** atp.Splitkey=ethIpProps This Attribute is only used by the AUTOSAR Classic Platform.
maximum Transmission Unit	PositiveInteger	0..1	attr	This attribute specifies the maximum transmission unit in bytes.
neighborCache Size	PositiveInteger	0..1	attr	This attribute specifies the size of neighbor cache or ARP table in units of entries.

Table B.19: EthernetCommunicationConnector

Class	«atpVariation» **EthernetCommunicationController**			
Note	Ethernet specific communication port attributes.			
Base	*ARObject*, *CommunicationController*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcuInstance.commController, MachineDesign.communicationController			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
canXlConfig	AbstractCan Communication Controller	0..1	ref	If the Ethernet frames handled by this Ethernet CommunicationController are to be tunneled through CAN XL, then this reference shall refer to the Abstract CanCommunicationController that aggregates the Can ControllerXlConfiguration of the physical CAN XL channel to be used for tunneling. **Tags:** atp.Splitkey=canXlConfig
couplingPort	CouplingPort	*	aggr	Optional CouplingPort that can be used to connect the ECU to a CouplingElement (e.g. a switch). **Tags:** atp.Splitkey=couplingPort, couplingPort.shortName
macLayerType	EthernetMacLayerType Enum	0..1	attr	Specifies the mac layer type of the ethernet controller.
macUnicast Address	MacAddressString	0..1	attr	Media Access Control address (MAC address) that uniquely identifies each EthernetCommunication Controller in the network.
maximum ReceiveBuffer Length	Integer	0..1	attr	Determines the maximum receive buffer length (frame length) in bytes.
maximum TransmitBuffer Length	Integer	0..1	attr	Determines the maximum transmit buffer length (frame length) in bytes.

▽

△

Class	«atpVariation» **EthernetCommunicationController**			
slaveActAs Passive Communication Slave	Boolean	0..1	attr	This attribute specifies if the EcuInstance is acting as a passive communication slave on the connected Physical Channel. This is used for EthernetCommunication Controllers that use Ethernet hardware which supports wake-up and sleep on the network (e.g. Open Alliance TC10 compliant Ethernet hardware).
slaveQualified UnexpectedLink DownTime	TimeValue	0..1	attr	This attribute specifies time when an unexpected link down is evaluated as link down and indicated to the AUTOSAR communication stack.

Table B.20: EthernetCommunicationController

Enumeration	**EthernetPhysicalLayerTypeEnum**
Note	Specifies physical layer types of Ethernet transceiver links.
Aggregated by	CouplingPort.physicalLayerType
Literal	***Description***
_10000BASE_T1	Ethernet Standard (IEEE 802.3ch) to support 10Gbit/s over a single twisted pair cable. **Tags:** atp.EnumerationLiteralIndex=13 xml.name=10000BASE-T1
_1000BASE_T	Ethernet Standard (IEEE 802.3ab) to support 1Gbit/s over 4 twisted pairs. **Tags:** atp.EnumerationLiteralIndex=6 xml.name=1000BASE-T
_1000BASE_T1	Ethernet Standard (IEEE 802.3bp) to support 1Gbit/s over a single twisted pair cable. **Tags:** atp.EnumerationLiteralIndex=8 xml.name=1000BASE-T1
_100BASE_T1	Ethernet Standard (IEEE 802.3bw) to support 100Mbit/s over a single twisted pair cable. 100 BASE-T1 is the IEEE Standardized version of BroadRReach. **Tags:** atp.EnumerationLiteralIndex=7 xml.name=100BASE-T1
_100BASE_TX	Ethernet Standard (IEEE 802.3u) to support 100Mbit/s over two twisted pairs. **Tags:** atp.EnumerationLiteralIndex=5 xml.name=100BASE-TX
_10BASE_T1S	Physical layer interface 10BASE-T1S (10Mbit/s, 2 pairs). Used for automotive. **Tags:** atp.EnumerationLiteralIndex=10 atp.Status=draft xml.name=10BASE-T1S
_2500BASE_T1	Ethernet Standard (IEEE 802.3ch) to support 2.5Gbit/s over a single twisted pair cable. **Tags:** atp.EnumerationLiteralIndex=11 xml.name=2500BASE-T1
_5000BASE_T1	Ethernet Standard (IEEE 802.3ch) to support 5Gbit/s over a single twisted pair cable. **Tags:** atp.EnumerationLiteralIndex=12 xml.name=5000BASE-T1

▽

△

Enumeration	**EthernetPhysicalLayerTypeEnum**
iEEE802_11P	Ethernet Standard (IEEE 802.11p) to support wireless communication in vehicular environments. **Tags:** atp.EnumerationLiteralIndex=9 xml.name=IEEE802-11P

Table B.21: EthernetPhysicalLayerTypeEnum

Class	**Gateway**			
Note	A gateway is an ECU that is connected to two or more clusters (channels, but not redundant), and performs a frame, Pdu or signal mapping between them. **Tags:** atp.recommendedPackage=Gateways			
Base	*ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ecu	EcuInstance	0..1	ref	Reference to one ECU instance that implements the gateway. **Tags:** atp.Splitkey=ecu
frameMapping	FrameMapping	*	aggr	Frame Gateway: The entire source frame is mapped as it is onto the target frame (what in general is only possible inside of a common platform). In this case source and target frame should be the identical object. atpVariation: If frames are variable in clusters, the gateway frame mapping needs to be variable, too. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=frameMapping, frameMapping.variation Point.shortLabel vh.latestBindingTime=postBuild
iPduMapping	IPduMapping	*	aggr	IPdu Gateway: Arranges those IPdus that are transferred by the gateway from one channel to the other in pairs and defines the mapping between them. atpVariation: If PDUs are variable in clusters, the gateway PDU mapping needs to be variable, too. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=iPduMapping, iPduMapping.variation Point.shortLabel vh.latestBindingTime=postBuild
signalMapping	ISignalMapping	*	aggr	Signal Gateway: Arranges those signals that are transferred by the gateway from one channel to the other in pairs and defines the mapping between them. atpVariation: If signals are variable in clusters, the gateway signal mapping needs to be variable, too. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=signalMapping, signalMapping.variation Point.shortLabel vh.latestBindingTime=postBuild

Table B.22: Gateway

Class	***IPdu*** (abstract)			
Note	The IPdu (Interaction Layer Protocol Data Unit) element is used to sum up all Pdus that are routed by the PduR.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Pdu*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Subclasses	ContainerIPdu, DcmIPdu, GeneralPurposeIPdu, ISignalIPdu, J1939DcmIPdu, J1939ProtectedIPdu, MultiplexedIPdu, NPdu, SecuredIPdu, UserDefinedIPdu			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table B.23: IPdu

Class	**IPduMapping**			
Note	Arranges those IPdus that are transferred by the gateway from one channel to the other in pairs and defines the mapping between them.			
Base	*ARObject*			
Aggregated by	Gateway.iPduMapping			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
introduction	DocumentationBlock	0..1	aggr	This represents introductory documentation about the IPdu mapping. **Tags:** atp.Splitkey=introduction
pduMaxLength	PositiveInteger	0..1	attr	Define the maximum length in bytes which limits the length of the Pdu during gateway operation if the runtime length of the received Pdu exceeds this limit.
pdurTpChunk Size	PositiveInteger	0..1	attr	Optionally defines the to be configured Pdu Router Tp ChunkSize for this routing relation.
sourceIPdu	PduTriggering	0..1	ref	Source destination of the referencing mapping. **Tags:** atp.Splitkey=sourceIPdu
targetIPdu	TargetIPduRef	0..1	aggr	Target destination of the referencing mapping. **Tags:** atp.Splitkey=targetIPdu

Table B.24: IPduMapping

Class	**ISignal**			
Note	Signal of the Interaction Layer. The RTE supports a "signal fan-out" where the same System Signal is sent in different SignalIPdus to multiple receivers. To support the RTE "signal fan-out" each SignalIPdu contains ISignals. If the same System Signal is to be mapped into several SignalIPdus there is one ISignal needed for each ISignalToIPduMapping. ISignals describe the Interface between the Precompile configured RTE and the potentially Postbuild configured Com Stack (see ECUC Parameter Mapping). In case of the SystemSignalGroup an ISignal shall be created for each SystemSignal contained in the SystemSignalGroup. **Tags:** atp.recommendedPackage=ISignals			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	**ISignal**			
data Transformation	DataTransformation	0..1	ref	Optional reference to a DataTransformation which represents the transformer chain that is used to transform the data that shall be placed inside this ISignal. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=dataTransformation, data Transformation.dataTransformation, data Transformation.variationPoint.shortLabel vh.latestBindingTime=codeGenerationTime
dataTypePolicy	DataTypePolicyEnum	0..1	attr	With the aggregation of SwDataDefProps an ISignal specifies how it is represented on the network. This representation follows a particular policy. Note that this causes some redundancy which is intended and can be used to support flexible development methodology as well as subsequent integrity checks. If the policy "networkRepresentationFromComSpec" is chosen the network representation from the ComSpec that is aggregated by the PortPrototype shall be used. If the "override" policy is chosen the requirements specified in the PortInterface and in the ComSpec are not fulfilled by the networkRepresentationProps. In case the System Description doesn't use a complete Software Component Description (VFB View) the "legacy" policy can be chosen.
initValue	ValueSpecification	0..1	aggr	Optional definition of a ISignal's initValue in case the System Description doesn't use a complete Software Component Description (VFB View). This supports the inclusion of legacy system signals. This value can be used to configure the Signal's "Init Value". If a full DataMapping exist for the SystemSignal this information may be available from a configured Sender ComSpec and ReceiverComSpec. In this case the initvalues in SenderComSpec and/or ReceiverComSpec override this optional value specification. Further restrictions apply from the RTE specification. **Tags:** atp.Splitkey=initValue
iSignalProps	ISignalProps	0..1	aggr	Additional optional ISignal properties that may be stored in different files. **Tags:** atp.Splitkey=iSignalProps
iSignalType	ISignalTypeEnum	0..1	attr	This attribute defines whether this iSignal is an array that results in a UINT8_N / UINT8_DYN ComSignalType in the COM configuration or a primitive type.
length	UnlimitedInteger	0..1	attr	Size of the signal in bits. The size needs to be derived from the mapped VariableDataPrototype according to the mapping of primitive DataTypes to BaseTypes as used in the RTE. Indicates maximum size for dynamic length signals. The ISignal length of zero bits is allowed.

▽

△

Class	**ISignal**			
network Representation Props	SwDataDefProps	0..1	aggr	Specification of the actual network representation. The usage of SwDataDefProps for this purpose is restricted to the attributes compuMethod and baseType. The optional baseType attributes "memAllignment" and "byteOrder" shall not be used. The attribute "dataTypePolicy" in the SystemTemplate element defines whether this network representation shall be ignored and the information shall be taken over from the network representation of the ComSpec. If "override" is chosen by the system integrator the network representation can violate against the requirements defined in the PortInterface and in the network representation of the ComSpec. In case that the System Description doesn't use a complete Software Component Description (VFB View) this element is used to configure "ComSignalDataInvalid Value" and the Data Semantics. **Tags:** atp.Splitkey=networkRepresentationProps
systemSignal	SystemSignal	0..1	ref	Reference to the System Signal that is supposed to be transmitted in the ISignal. **Tags:** atp.Splitkey=systemSignal
timeout Substitution Value	ValueSpecification	0..1	aggr	Defines and enables the ComTimeoutSubstituition for this ISignal. **Tags:** atp.Splitkey=timeoutSubstitutionValue
transformation ISignalProps	TransformationISignal Props	*	aggr	A transformer chain consists of an ordered list of transformers. The ISignal specific configuration properties for each transformer are defined in the TransformationISignalProps class. The transformer configuration properties that are common for all ISignals are described in the TransformationTechnology class. **Tags:** atp.Splitkey=transformationISignalProps

Table B.25: ISignal

Class	**ISignalIPdu**			
Note	Represents the IPdus handled by Com. The ISignalIPdu assembled and disassembled in AUTOSAR COM consists of one or more signals. In case no multiplexing is performed this IPdu is routed to/from the Interface Layer. A maximum of one dynamic length signal per IPdu is allowed. **Tags:** atp.recommendedPackage=Pdus			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *IPdu*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Pdu*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**ISignalIPdu**			
iPduTiming Specification	IPduTiming	0..1	aggr	Timing specification for Com IPdus (Transmission Modes). This information is mandatory for the sender in a System Extract. This information may be omitted on receivers in a System Extract. atpVariation: The timing of a Pdu can vary. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=iPduTimingSpecification, iPduTiming Specification.variationPoint.shortLabel vh.latestBindingTime=postBuild
iSignalToPdu Mapping	ISignalToIPduMapping	*	aggr	Definition of SignalToIPduMappings included in the Signal IPdu. atpVariation: The content of a PDU can be variable. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=iSignalToPduMapping, iSignalToPdu Mapping.shortName, iSignalToPduMapping.variation Point.shortLabel vh.latestBindingTime=postBuild
unusedBit Pattern	Integer	0..1	attr	AUTOSAR COM and AUTOSAR IPDUM are filling not used areas of an IPDU with this bit-pattern. This attribute is mandatory to avoid undefined behavior. This byte-pattern will be repeated throughout the IPdu.

Table B.26: ISignalIPdu

Class	**ISignalPort**			
Note	Connectors reception or send port on the referenced channel referenced by an ISignalTriggering. If different timeouts or DataFilters for ISignals need to be specified several ISignalPorts may be created.			
Base	*ARObject*, *CommConnectorPort*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*CommunicationConnector*.ecuCommPortInstance			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataFilter	DataFilter	0..1	aggr	Optional specification of a signal COM filter at the receiver side in case that the System Description doesn't use a complete Software Component Description (VFB View). This supports the inclusion of legacy system signals. If a full DataMapping exist for the SystemSignal this information may be available from a configured ReceiverComSpec. In this case the ReceiverComSpec overrides this optional specification. **Tags:** atp.Splitkey=dataFilter
ddsQosProfile	DdsCpQosProfile	0..1	ref	Reference to the DDS Qos profile used for this ISignal. **Tags:** atp.Splitkey=ddsQosProfile atp.Status=candidate
firstTimeout	TimeValue	0..1	attr	• ISignalPort with communicationDirection = in: Optional first timeout value in seconds for the reception of the ISignal. • ISignalPort with communicationDirection = out: Optional first timeout value in seconds for transmission deadline monitoring.
handleInvalid	HandleInvalidEnum	0..1	attr	This attribute defines how invalidation is applied to the ISignals received in the context of this ISignalPort.

▽

△

Class	**ISignalPort**			
timeout	TimeValue	0..1	attr	• ISignalPort with communicationDirection = in: Optional timeout value in seconds for the reception of the ISignal. The attribute value is used to configure the Com Timeout in the COM module. The RTE ignores this attribute. The timeout can also be specified with the NonqueuedReceiverComSpec.aliveTimeout attribute. If a full DataMapping exists for the SystemSignal and the value is available in the configured ReceiverComSpec, then the timeout value in the ReceiverComSpec overrides this optional timeout specification during the creation of the Base Ecu Configuration of the COM module. • ISignalPort with communicationDirection = out: Optional timeout value in seconds for the transmission of the ISignal. The attribute value is used to configure the ComTimeout in the COM module. The RTE ignores this attribute. The timeout can also be specified with the SenderComSpec.transmissionAcknowledge.timeout attribute. If a full DataMapping exists for the System Signal and the value is available in the configured Sender ComSpec, then the timeout value in the SenderComSpec overrides this optional timeout specification during the creation of the Base Ecu Configuration of the COM module. This attribute can be used in the following cases: • legacy signal where the System Description doesn't use a complete Software Component Description (VFB View) and where the DataMapping is missing. • bus monitoring use cases in which the DataMapping is ignored.

Table B.27: ISignalPort

Class	**ISignalTriggering**			
Note	A ISignalTriggering allows an assignment of ISignals to physical channels.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*PhysicalChannel*.iSignalTriggering			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
iSignal	ISignal	0..1	ref	This reference shall be used if an ISignal is transported on the PhysicalChannel. This reference forms an XOR relationship with the ISignalTriggering-ISignalGroup reference. **Tags:** atp.Splitkey=iSignal
iSignalGroup	ISignalGroup	0..1	ref	This reference shall be used if an ISignalGroup is transported on the PhysicalChannel. This reference forms an XOR relationship with the ISignal Triggering-ISignal reference. **Tags:** atp.Splitkey=iSignalGroup
iSignalPort	ISignalPort	*	ref	References to the ISignalPort on every ECU of the system which sends and/or receives the ISignal. References for both the sender and the receiver side shall be included when the system is completely defined. **Tags:** atp.Splitkey=iSignalPort

Table B.28: ISignalTriggering

Class	PPortPrototype			
Note	Component port providing a certain port interface.			
Base	*ARObject*, *AbstractProvidedPortPrototype*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
provided Interface	PortInterface	0..1	tref	The interface that this port provides. **Stereotypes:** isOfType **Tags:** atp.Splitkey=providedInterface

Table B.29: PPortPrototype

Class	PduTriggering			
Note	The PduTriggering describes on which channel the IPdu is transmitted. The Pdu routing by the PduR is only allowed for subclasses of IPdu. Depending on its relation to entities such channels and clusters it can be unambiguously deduced whether a fan-out is handled by the Pdu router or the Bus Interface. If the fan-out is specified between different clusters it shall be handled by the Pdu Router. If the fan-out is specified between different channels of the same cluster it shall be handled by the Bus Interface.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*PhysicalChannel*.pduTriggering			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
iPdu	Pdu	0..1	ref	Reference to the Pdu for which the PduTriggering is defined. One I-Pdu can be triggered on different channels (PduR fan-out). The Pdu routing by the PduR is only allowed for subclasses of IPdu. Nevertheless is the reference to the Pdu element necessary since the PduTriggering element is also used to specify the sending and receiving connections to Ecu Ports. **Tags:** atp.Splitkey=iPdu
iPduPort	IPduPort	*	ref	References to the IPduPort on every ECU of the system which sends and/or receives the I-PDU. References for both the sender and the receiver side shall be included when the system is completely defined. **Tags:** atp.Splitkey=iPduPort
iSignal Triggering	ISignalTriggering	*	ref	This reference provides the relationship to the ISignal Triggerings that are implemented by the PduTriggering. The reference is optional since no ISignalTriggering can be defined for DCM and Multiplexed Pdus. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=iSignalTriggering, iSignalTriggering.iSignal Triggering, iSignalTriggering.variationPoint.shortLabel vh.latestBindingTime=postBuild
secOcCrypto Mapping	SecOcCryptoService Mapping	0..1	ref	This reference identifies the crypto profile applicable to the usage (send, receive) of the also referenced Secured IPdu. Obviously, this reference is only applicable if the Pdutriggering also references a SecuredIPdu in the role i Pdu. **Tags:** atp.Splitkey=secOcCryptoMapping atp.Status=obsolete



△

Class	**PduTriggering**			
secOcCrypto ServiceMapping	SecOcCryptoService Mapping	0..1	ref	This reference identifies the crypto profile applicable to the usage (send, receive) of the also referenced Secured IPdu. Obviously, this reference is only applicable if the Pdutriggering also references a SecuredIPdu in the role i Pdu. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=secOcCryptoServiceMapping, secOcCrypto ServiceMapping.secOcCryptoServiceMapping, secOc CryptoServiceMapping.variationPoint.shortLabel vh.latestBindingTime=postBuild
triggerIPduSend Condition	TriggerIPduSend Condition	*	aggr	Defines the trigger for the Com_TriggerIPDUSend API call. Only if all defined TriggerIPduSendConditions evaluate to true (AND associated) the Com_Trigger IPDUSend API shall be called. **Tags:** atp.Splitkey=triggerIPduSendCondition

Table B.30: PduTriggering

Class	***PhysicalChannel*** (abstract)			
Note	A physical channel is the transmission medium that is used to send and receive information between communicating ECUs. Each CommunicationCluster has at least one physical channel. Bus systems like CAN and LIN only have exactly one PhysicalChannel. A FlexRay cluster may have more than one PhysicalChannels that may be used in parallel for redundant communication. An ECU is part of a cluster if it contains at least one controller that is connected to at least one channel of the cluster.#			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*AbstractCanPhysicalChannel*, EthernetPhysicalChannel, FlexrayPhysicalChannel, LinPhysicalChannel, UserDefinedPhysicalChannel			
Aggregated by	*CommunicationCluster*.physicalChannel			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
comm Connector	Communication Connector	*	ref	Reference to the ECUInstance via a Communication Connector to which the channel is connected. atpVariation: Variable assignment of Physical Channels to different CommunicationConnectors is expressed with this variation. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=commConnector, comm Connector.communicationConnector, comm Connector.variationPoint.shortLabel vh.latestBindingTime=postBuild This Attribute is only used by the AUTOSAR Classic Platform.

▽

△

Class	***PhysicalChannel*** (abstract)			
frameTriggering	FrameTriggering	*	aggr	One frame triggering is defined for exactly one channel. Channels may have assigned an arbitrary number of frame triggerings. atpVariation: If signals/PDUs/frames are variable, the corresponding triggerings shall be variable, too. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=frameTriggering, frameTriggering.shortName, frameTriggering.variationPoint.shortLabel vh.latestBindingTime=postBuild This Attribute is only used by the AUTOSAR Classic Platform.
iSignal Triggering	ISignalTriggering	*	aggr	One ISignalTriggering is defined for exactly one channel. Channels may have assigned an arbitrary number of ISignaltriggerings. atpVariation: If signals/PDUs/frames are variable, the corresponding triggerings shall be variable, too. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=iSignalTriggering, iSignalTriggering.shortName, iSignalTriggering.variationPoint.shortLabel vh.latestBindingTime=postBuild This Attribute is only used by the AUTOSAR Classic Platform.
managed Physical Channel	PhysicalChannel	*	ref	Reference between a channel with role managing channel and a channel with role managed channel. **Tags:** atp.Splitkey=managedPhysicalChannel This Attribute is only used by the AUTOSAR Classic Platform.
pduTriggering	PduTriggering	*	aggr	One PduTriggering is defined for exactly one channel. Channels may have assigned an arbitrary number of I-Pdu triggerings. atpVariation: If signals/PDUs/frames are variable, the corresponding triggerings shall be variable, too. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=pduTriggering, pduTriggering.shortName, pduTriggering.variationPoint.shortLabel vh.latestBindingTime=postBuild This Attribute is only used by the AUTOSAR Classic Platform.

Table B.31: PhysicalChannel

Class	***PortPrototype*** (abstract)			
Note	Base class for the ports of an AUTOSAR software component. The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*AbstractProvidedPortPrototype*, *AbstractRequiredPortPrototype*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	***PortPrototype*** (abstract)			
clientServer Annotation	ClientServerAnnotation	*	aggr	Annotation of this `PortPrototype` with respect to client/ server communication. **Tags:** atp.Splitkey=clientServerAnnotation
delegatedPort Annotation	DelegatedPort Annotation	0..1	aggr	Annotations on this delegated port. **Tags:** atp.Splitkey=delegatedPortAnnotation
ioHwAbstraction Server Annotation	IoHwAbstractionServer Annotation	*	aggr	Annotations on this IO Hardware Abstraction port. **Tags:** atp.Splitkey=ioHwAbstractionServerAnnotation
modePort Annotation	ModePortAnnotation	*	aggr	Annotations on this mode port. **Tags:** atp.Splitkey=modePortAnnotation
nvDataPort Annotation	NvDataPortAnnotation	*	aggr	Annotations on this non voilatile data port. **Tags:** atp.Splitkey=nvDataPortAnnotation
parameterPort Annotation	ParameterPort Annotation	*	aggr	Annotations on this parameter port. **Tags:** atp.Splitkey=parameterPortAnnotation
senderReceiver Annotation	SenderReceiver Annotation	*	aggr	Collection of annotations of this ports sender/receiver communication. **Tags:** atp.Splitkey=senderReceiverAnnotation
triggerPort Annotation	TriggerPortAnnotation	*	aggr	Annotations on this trigger port. **Tags:** atp.Splitkey=triggerPortAnnotation

Table B.32: PortPrototype

Class	**RPortPrototype**			
Note	Component port requiring a certain port interface.			
Base	*ARObject*, *AbstractRequiredPortPrototype*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
mayBe Unconnected	Boolean	0..1	attr	If set to true, this attribute indicates that the enclosing `RPortPrototype` may be left unconnected and that this aspect has explicitly been considered in the software-component's design. This Attribute is only used by the AUTOSAR Classic Platform.
required Interface	PortInterface	0..1	tref	The interface that this port requires. **Stereotypes:** isOfType **Tags:** atp.Splitkey=requiredInterface

Table B.33: RPortPrototype

Class	**RootSwCompositionPrototype**			
Note	The RootSwCompositionPrototype represents the top-level-composition of software components within a given System. According to the use case of the System, this may for example be a more or less complete VFB description, the software of a System Extract or the software of a flat ECU Extract with only atomic SWCs. Therefore the RootSwComposition will only occasionally contain all atomic software components that are used in a complete VFB System. The OEM is primarily interested in the required functionality and the interfaces defining the integration of the Software Component into the System. The internal structure of such a component contains often substantial intellectual property of a supplier. Therefore a top-level software composition will often contain empty compositions which represent subsystems. The contained SwComponentPrototypes are fully specified by their SwComponentTypes (including Port Prototypes, PortInterfaces, VariableDataPrototypes, SwcInternalBehavior etc.), and their ports are interconnected using SwConnectorPrototypes.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, System.rootSoftwareComposition			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
calibration ParameterValue Set	CalibrationParameter ValueSet	*	ref	Used CalibrationParameterValueSet for instance specific initialization of calibration parameters. **Tags:** atp.Splitkey=calibrationParameterValueSet This Attribute is only used by the AUTOSAR Classic Platform.
flatMap	FlatMap	0..1	ref	The FlatMap used in the scope of this RootSw CompositionPrototype. **Tags:** atp.Splitkey=flatMap This Attribute is only used by the AUTOSAR Classic Platform.
software Composition	CompositionSw ComponentType	0..1	tref	We assume that there is exactly one top-level composition that includes all Component instances of the system. **Stereotypes:** isOfType **Tags:** atp.Splitkey=softwareComposition

Table B.34: RootSwCompositionPrototype

Class	**SenderReceiverToSignalMapping**			
Note	Mapping of a sender receiver communication data element to a signal.			
Base	*ARObject*, *DataMapping*			
Aggregated by	SystemMapping.dataMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataElement	VariableDataPrototype	0..1	iref	Reference to the data element. **Tags:** atp.Splitkey=dataElement, dataElement.context Component, dataElement.contextComposition, data Element.contextPort, dataElement.targetDataPrototype **InstanceRef implemented by:** VariableDataPrototypeIn SystemInstanceRef
senderToSignal TextTable Mapping	TextTableMapping	0..1	aggr	This mapping allows for the text-table translation between the sending DataPrototype that is defined in the Port Prototype and the physicalProps defined for the System Signal. **Tags:** atp.Splitkey=senderToSignalTextTableMapping
signalTo ReceiverText TableMapping	TextTableMapping	0..1	aggr	This mapping allows for the text-table translation between the physicalProps defined for the SystemSignal and a receiving DataPrototype that is defined in the Port Prototype. **Tags:** atp.Splitkey=signalToReceiverTextTableMapping

△

Class	**SenderReceiverToSignalMapping**			
systemSignal	SystemSignal	0..1	ref	Reference to the system signal used to carry the data element. **Tags:** atp.Splitkey=systemSignal

Table B.35: SenderReceiverToSignalMapping

Class	**SensorActuatorSwComponentType**			
Note	The SensorActuatorSwComponentType introduces the possibility to link from the software representation of a sensor/actuator to its hardware description provided by the ECU Resource Template. **Tags:** atp.recommendedPackage=SwComponentTypes			
Base	*ARElement*, *ARObject*, *AtomicSwComponentType*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *Collectable Element*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
sensorActuator	HwDescriptionEntity	0..1	ref	Reference from the Sensor Actuator Software Component Type to the description of the actual hardware. **Tags:** atp.Splitkey=sensorActuator

Table B.36: SensorActuatorSwComponentType

Class	***SwComponentType*** (abstract)			
Note	Base class for AUTOSAR software components.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*AtomicSwComponentType*, CompositionSwComponentType, ParameterSwComponentType			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
consistency Needs	ConsistencyNeeds	*	aggr	This represents the collection of `ConsistencyNeeds` owned by the enclosing `SwComponentType`. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=consistencyNeeds, consistencyNeeds.shortName, consistencyNeeds.variationPoint.shortLabel vh.latestBindingTime=preCompileTime This Attribute is only used by the AUTOSAR Classic Platform.
port	PortPrototype	*	aggr	The `PortPrototype`s through which this `SwComponentType` can communicate. The aggregation of `PortPrototype` is subject to variability with the purpose to support the conditional existence of `PortPrototype`s. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=port, port.shortName, port.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

▽

△

Class	***SwComponentType*** (abstract)			
portGroup	PortGroup	*	aggr	A port group being part of this component. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=portGroup, portGroup.shortName, port Group.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
swcMapping Constraint	SwComponentMapping Constraints	*	ref	Reference to constraints that are valid for this Sw ComponentType. **Tags:** atp.Splitkey=swcMappingConstraint This Attribute is only used by the AUTOSAR Classic Platform.
swComponent Documentation	SwComponent Documentation	0..1	aggr	This adds a documentation to the `SwComponentType`. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=swComponentDocumentation, sw ComponentDocumentation.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=-10
unitGroup	UnitGroup	*	ref	This allows for the specification of which `UnitGroup`s are relevant in the context of referencing `SwComponentType`. **Tags:** atp.Splitkey=unitGroup This Attribute is only used by the AUTOSAR Classic Platform.

Table B.37: SwComponentType

Class	**System**			
Note	The top level element of the System Description. The System description defines five major elements: Topology, Software, Communication, Mapping and Mapping Constraints. The System element directly aggregates the elements describing the Software, Mapping and Mapping Constraints; it contains a reference to an ASAM FIBEX description specifying Communication and Topology. **Tags:** atp.recommendedPackage=Systems			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element, *AtpClassifier*.atpFeature			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
clientId DefinitionSet	ClientIdDefinitionSet	*	ref	Set of Client Identifiers that are used for inter-ECU client-server communication in the System. **Tags:** atp.Splitkey=clientIdDefinitionSet This Attribute is only used by the AUTOSAR Classic Platform.
containerIPdu HeaderByte Order	ByteOrderEnum	0..1	attr	Defines the byteOrder of the header in ContainerIPdus. This Attribute is only used by the AUTOSAR Classic Platform.
ecuExtract Version	RevisionLabelString	0..1	attr	Version number of the Ecu Extract. This Attribute is only used by the AUTOSAR Classic Platform.

▽

△

Class	System			
fibexElement	FibexElement	*	ref	Reference to ASAM FIBEX elements specifying Communication and Topology. All Fibex Elements used within a System Description shall be referenced from the System Element. atpVariation: In order to describe a product-line, all Fibex Elements can be optional. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=fibexElement, fibexElement.fibexElement, fibexElement.variationPoint.shortLabel vh.latestBindingTime=postBuild
interpolation Routine MappingSet	InterpolationRoutine MappingSet	*	ref	This reference identifies the InterpolationRoutineMapping Sets that are relevant in the context of the enclosing System. **Tags:** atp.Splitkey=interpolationRoutineMappingSet This Attribute is only used by the AUTOSAR Classic Platform.
j1939Shared AddressCluster	J1939SharedAddress Cluster	*	aggr	Collection of J1939Clusters that share a common address space for the routing of messages. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=j1939SharedAddressCluster, j1939Shared AddressCluster.shortName, j1939SharedAddress Cluster.variationPoint.shortLabel vh.latestBindingTime=postBuild This Attribute is only used by the AUTOSAR Classic Platform.
mapping	SystemMapping	*	aggr	Aggregation of all mapping aspects (mapping of SW components to ECUs, mapping of data elements to signals, and mapping constraints). In order to support OEM / Tier 1 interaction and shared development for one common System this aggregation is atpSplitable and atpVariation. The content of System Mapping can be provided by several parties using different names for the SystemMapping. This element is not required when the System description is used for a network-only use-case. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=mapping, mapping.shortName, mapping.variationPoint.shortLabel vh.latestBindingTime=postBuild
pncVector Length	PositiveInteger	0..1	attr	Length of the partial networking request release information vector (in bytes).
pncVectorOffset	PositiveInteger	0..1	attr	Absolute offset (with respect to the NM-PDU) of the partial networking request release information vector that is defined in bytes as an index starting with 0.

▽

△

Class	System			
rootSoftware Composition	RootSwComposition Prototype	0..1	aggr	Aggregation of the root software composition, containing all software components in the System in a hierarchical structure. This element is not required when the System description is used for a network-only use-case. atpVariation: The RootSwCompositionPrototype can vary. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=rootSoftwareComposition, rootSoftware Composition.shortName, rootSoftware Composition.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime This Attribute is only used by the AUTOSAR Classic Platform.
swCluster	CpSoftwareCluster	*	ref	CP Software Clusters of this System **Stereotypes:** atpVariation **Tags:** atp.Splitkey=swCluster, swCluster.cpSoftwareCluster, sw Cluster.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime This Attribute is only used by the AUTOSAR Classic Platform.
systemCom SpecDefinition	SystemComSpec DefinitionSet	*	ref	Reference to the set of ComSpec definitions that are used for inter-ECU communication in the System. **Tags:** atp.Splitkey=systemComSpecDefinition
system Documentation	Chapter	*	aggr	Possibility to provide additional documentation while defining the System. The System documentation can be composed of several chapters. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=systemDocumentation, system Documentation.shortName, system Documentation.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime xml.sequenceOffset=-10 This Attribute is only used by the AUTOSAR Classic Platform.
systemVersion	RevisionLabelString	0..1	attr	Version number of the System Description.

Table B.38: System